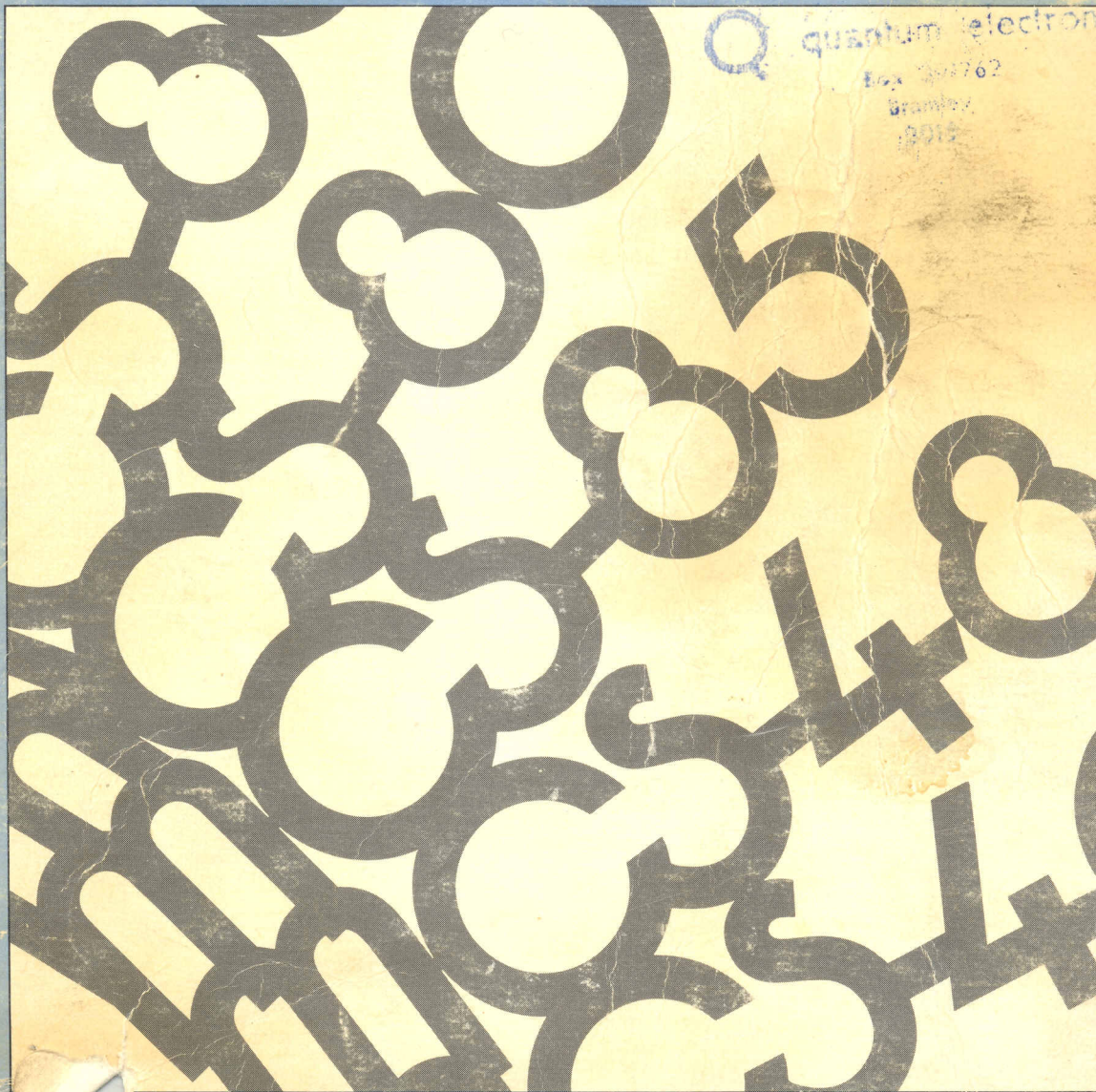


intel<sup>®</sup>

# PERIPHERAL DESIGN HANDBOOK

April 1978



on 1978

9800676A

\$5.00

This is the first edition of the PERIPHERAL DESIGN HANDBOOK. It contains data and applications information about the world's most complete line of microprocessor peripheral devices. Intel® is committed to providing the broadest line of high capability peripherals for the industry's leading microprocessor families, the MCS-48™, MCS-80™, and MCS-85™.

MC350™ and MC250™  
for the industry's leading microprocessor families, the MC348™  
to provide the broadest line of high-capacity peripheral devices. Intel's com-  
plete line of microprocessor peripheral devices, Intel's com-  
plete data and application information about the world's most  
popular is the first edition of the PERIPHERAL DESIGN HANDBOOK.

## Table of Contents

---

### Section 1

#### Peripheral Data Sheets

8041/8741 .....	1-1
Pin Description .....	1-4
UPI Instruction Set .....	1-5
8205 High Speed 1 Out of 8 Binary Decoder .....	1-8
Functional Description .....	1-9
Applications .....	1-10
8212 Eight-Bit Input/Output Port .....	1-14
Functional Description .....	1-15
Applications .....	1-16
8251A Programmable Communication Interface .....	1-24
Features and Enhancements .....	1-25
Functional Description .....	1-26
Applications .....	1-33
Timing Diagrams .....	1-37
8253, 8253-5 Programmable Interval Timer .....	1-40
Functional Description .....	1-41
Operational Description .....	1-43
8255A, 8255A-5 Programmable Peripheral Interface .....	1-51
Functional Description .....	1-52
Operational Description .....	1-54
Applications .....	1-65
Timing Diagrams .....	1-69
8257, 8257-5 Programmable DMA Controller .....	1-72
Functional Description .....	1-73
Operational Summary .....	1-79
Timing Diagrams .....	1-86
Applications .....	1-88
8259, 8259-5 Programmable Interrupt Controller .....	1-89
Functional Description .....	1-91
Detailed Operation Description .....	1-93
8259 Instruction Set .....	1-100
Timing Diagrams .....	1-103
8271 Programmable Floppy Disk Controller .....	1-104
Functional Description .....	1-106
Principles of Operation .....	1-114
Timing Diagrams .....	1-128
8273 Programmable HDLC/SDLC Controller .....	1-131
Functional Description .....	1-133
Principles of Operation .....	1-140
Timing Diagrams .....	1-154
8275 Programmable CRT Controller .....	1-156
Functional Description .....	1-158
System Operation .....	1-160
Timing Diagrams .....	1-177

---

## Table of Contents (CONT'D)

8278 Programmable Keyboard Interface .....	1-180
Pin Description .....	1-181
Command Summary .....	1-183
Timing Diagrams .....	1-189
8279, 8279-5 Programmable Keyboard Display Interface .....	1-190
Functional Description .....	1-191
Timing Diagrams .....	1-200
8291 GPIB Talker/Listener .....	1-201
8292 GPIB Controller .....	1-203
8294 Data Encryption Unit .....	1-204
8295 Dot Matrix Printer Controller .....	1-208

### Section II

#### Application Notes

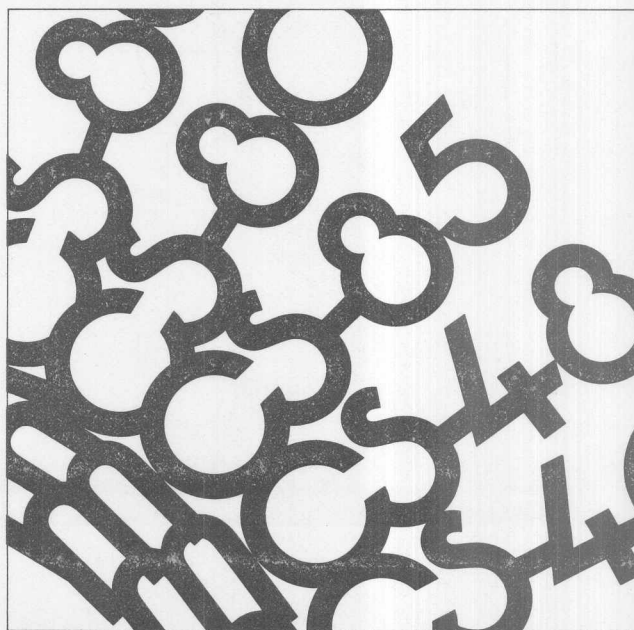
Printer Control with the UPI-41™ .....	2-2
Using the 8251 Universal Synchronous/Asynchronous Receiver/Transmitter .....	2-32
8255A Programmable Peripheral Interface Applications .....	2-63
Using the 8259 Programmable Interrupt Controller .....	2-93
CRT Terminal Design Using the Intel 8275 and 8279 .....	2-119

### Appendix 1

#### Article Reprints

Slave Microcomputer Lightens Main Microprocessor Load .....	3-3
Microcomputer Interfacing: Characteristics of the 8253 Programmable Interval Timer .....	3-7

**SECTION 1**  
**PERIPHERAL**  
**DATA SHEETS**







# 8041/8741 UNIVERSAL PERIPHERAL INTERFACE 8-BIT MICROCOMPUTER

**PRELIMINARY**  
Notice: This is not a final specification. Some parametric limits are subject to change.

- Fully Compatible with MCS-80™, MCS-85™ and MCS-48™ Microprocessor Families
- Single Level Interrupt
- 8-Bit CPU plus ROM, RAM, I/O, Timer and Clock in a Single Package
- Single 5V Supply
- Alternative to Custom LSI
- Pin Compatible ROM and EPROM Versions
- 1K x 8 ROM/EPROM, 64 x 8 RAM, 18 Programmable I/O Pins
- Asynchronous Data Register for Interface to Master Processor
- Expandable I/O

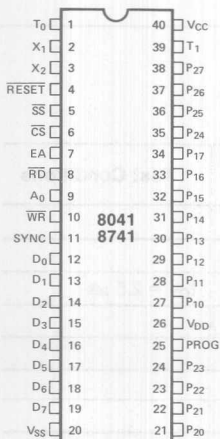
The Intel® 8041/8741 is a general purpose, programmable interface device designed for use with a variety of 8-bit microprocessor systems. It contains a low cost microcomputer with program memory, data memory, 8-bit CPU, I/O ports, timer/counter, and clock in a single 40-pin package. Interface registers are included to enable the UPI device to function as a peripheral controller in MCS-80™, MCS-85™, MCS-48™, and other 8-bit systems.

The UPI-41™ has 1K words of program memory and 64 words of data memory on-chip. To allow full user flexibility the program memory is available as ROM in the 8041 version or as UV-erasable EPROM in the 8741 version. The 8741 and the 8041 are fully pin compatible for easy transition from prototype to production level designs.

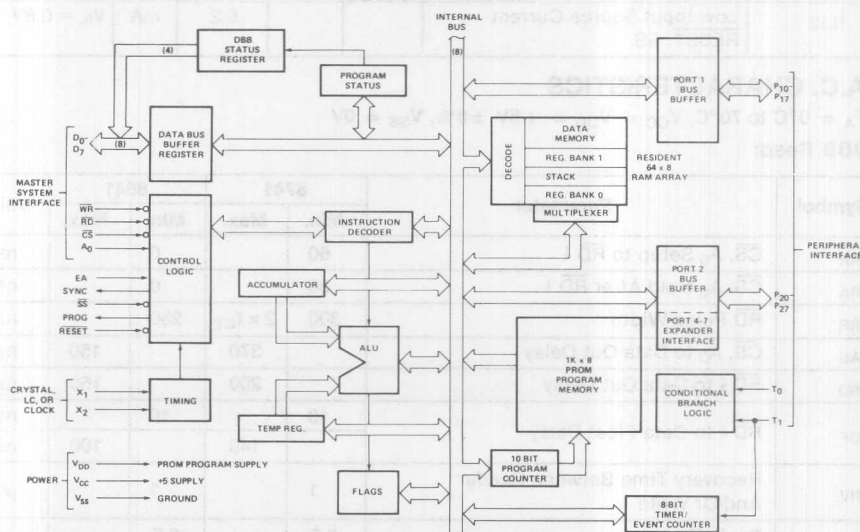
The device has two 8-bit, TTL compatible I/O ports and two test inputs. Individual port lines can function as either inputs or outputs under software control. I/O can be expanded with the 8243 device which is directly compatible and has 16 I/O lines. An 8-bit programmable timer/counter is included in the UPI device for generating timing sequences or counting external inputs. Additional UPI features include: single 5V supply, low power standby mode (in the 8041), single-step mode for debug (in the 8741), single level interrupt, and dual working register banks.

Because it's a complete microcomputer, the UPI provides more flexibility for the designer than conventional LSI interface devices. It is designed to be an efficient controller as well as an arithmetic processor. Applications include keyboard scanning, printer control, display multiplexing and similar functions which involve interfacing peripheral devices to microprocessor systems.

## PIN CONFIGURATION



## BLOCK DIAGRAM



**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature Under Bias ..... 0°C to 70°C  
 Storage Temperature ..... -65°C to +150°C  
 Voltage on Any Pin With Respect to Ground ..... -0.5V to +7V  
 Power Dissipation ..... 1.5 Watt

\*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

**D.C. AND OPERATING CHARACTERISTICS**

T<sub>A</sub> = 0°C to 70°C, V<sub>CC</sub> = V<sub>DD</sub> = +5V ± 5%, V<sub>SS</sub> = 0V

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
V <sub>IL</sub>	Input Low Voltage (All Except X <sub>1</sub> , X <sub>2</sub> )	-0.5		0.8	V	
V <sub>IH</sub>	Input High Voltage (All Except X <sub>1</sub> , X <sub>2</sub> RESET)	2.0		V <sub>CC</sub>	V	
V <sub>IH2</sub>	Input High Voltage (X <sub>1</sub> , RESET)	3.0		V <sub>CC</sub>	V	
V <sub>OL</sub>	Output Low Voltage (D <sub>0</sub> -D <sub>7</sub> , Sync)			0.45	V	I <sub>OL</sub> = 2.0 mA
V <sub>OL2</sub>	Output Low Voltage (All Other Outputs Except Prog)			0.45	V	I <sub>OL</sub> = 1.6 mA
V <sub>OH</sub>	Output High Voltage (D <sub>0</sub> -D <sub>7</sub> )	2.4			V	I <sub>OH</sub> = -400 μA
V <sub>OH1</sub>	Output High Voltage (All Other Outputs)	2.4			V	I <sub>OH</sub> = -50 μA
I <sub>IL</sub>	Input Leakage Current (T <sub>0</sub> , T <sub>1</sub> , RD, WR, CS, A <sub>0</sub> , EA)			±10	μA	V <sub>SS</sub> ≤ V <sub>IN</sub> ≤ V <sub>CC</sub>
I <sub>OL</sub>	Output Leakage Current (D <sub>0</sub> -D <sub>7</sub> , High Z State)			±10	μA	V <sub>SS</sub> + 0.45 ≤ V <sub>IN</sub> ≤ V <sub>CC</sub>
I <sub>DD</sub>	V <sub>DD</sub> Supply Current		10	25	mA	
I <sub>CC</sub> + I <sub>DD</sub>	Total Supply Current		65	135	mA	
V <sub>OL3</sub>	Output Low Voltage (Prog)			0.45	V	I <sub>OL</sub> = 1.0 mA
I <sub>L11</sub>	Low Input Source Current P <sub>10</sub> -P <sub>17</sub> P <sub>20</sub> -P <sub>27</sub>			0.4	mA	V <sub>IL</sub> = 0.8V
I <sub>L12</sub>	Low Input Source Current RESET, SS			0.2	mA	V <sub>IL</sub> = 0.8V

**A.C. CHARACTERISTICS**

T<sub>A</sub> = 0°C to 70°C, V<sub>CC</sub> = V<sub>DD</sub> = +5V ± 5%, V<sub>SS</sub> = 0V

**DBB Read:**

Symbol	Parameter	8741		8041		Units	Test Conditions
		Min.	Max.	Min.	Max.		
t <sub>AR</sub>	CS, A <sub>0</sub> Setup to RD ↓	60		0		ns	
t <sub>RA</sub>	CS, A <sub>0</sub> Hold After RD ↑	30		0		ns	
t <sub>RR</sub>	RD Pulse Width	300	2 × t <sub>CY</sub>	250		ns	t <sub>CY</sub> = 2.5 μs
t <sub>AD</sub>	CS, A <sub>0</sub> to Data Out Delay		370		150	ns	
t <sub>RD</sub>	RD ↓ to Data Out Delay		200		150	ns	
t <sub>DF</sub>	RD ↑ to Data Float Delay	10		10		ns	
			140		100	ns	
t <sub>RV</sub>	Recovery Time Between Reads And/Or Write	1		1		μs	
t <sub>CY</sub>	Cycle Time	2.5		2.5		μs	6 MHz Crystal

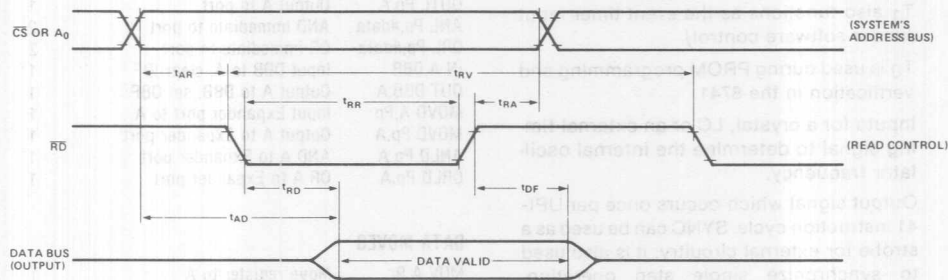
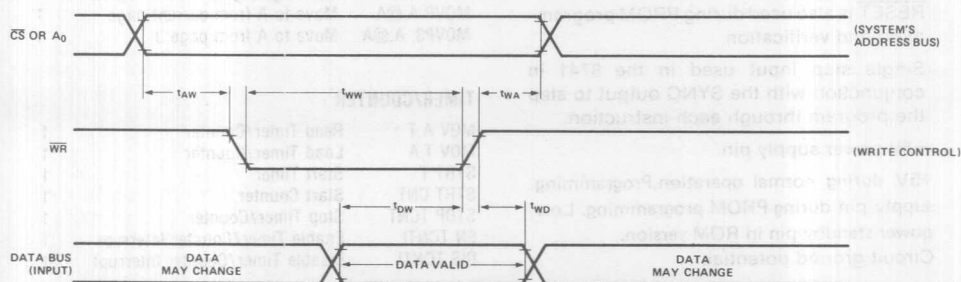
**PRELIMINARY**  
 Notice: This is not a final specification. Some parametric limits are subject to change.

**DBB Write:**

Symbol	Parameter	8741		8041		Units	Test Conditions
		Min.	Max.	Min.	Max.		
$t_{AW}$	$\overline{CS}$ , $A_0$ Setup to $\overline{WR} \downarrow$	60		0		ns	
$t_{WA}$	$\overline{CS}$ , $A_0$ Hold After $\overline{WR} \uparrow$	30		0		ns	
$t_{WW}$	$\overline{WR}$ Pulse Width	300	$2 \times t_{CY}$	250		ns	$t_{CY} = 2.5 \mu s$
$t_{DW}$	Data Setup to $\overline{WR} \uparrow$	250		150		ns	
$t_{WD}$	Data Hold After $\overline{WR} \uparrow$	30		0		ns	

**A.C. TEST CONDITIONS**

D7-D0 Outputs

 $R_L = 2.2k \text{ to } V_{SS}$  $4.3k \text{ to } V_{CC}$  $C_L = 100 \text{ pF}$ **WAVEFORMS****Read Operation — Data Bus Buffer Register****Write Operation — Data Bus Buffer Register**

## PIN DESCRIPTION

Signal	Description
D <sub>0</sub> -D <sub>7</sub>	Three-state, bi-directional, DATA BUS BUFFER lines used to interface the UPI-41 to an 8-bit master system data bus.
P <sub>10</sub> -P <sub>17</sub>	8-bit, PORT 1, quasi-bi-directional I/O lines.
P <sub>20</sub> -P <sub>27</sub>	8-bit, PORT 2, quasi-bi-directional I/O lines The lower 4-bits (P <sub>20</sub> -P <sub>23</sub> ) interface directly to the 8243 I/O expander device and contain address and data information during PORT 4-7 access.
$\overline{WR}$	I/O write input which enables the master CPU to write data and command words to the UPI-41 DATA BUS BUFFER.
$\overline{RD}$	I/O read input which enables the master CPU to read data and status words from the DATA BUS BUFFER or status register.
$\overline{CS}$	Chip select input used to select one UPI-41 out of several connected to a common data bus.
A <sub>0</sub>	Address input used by the master processor to indicate whether byte transfer is data or command.
T <sub>0</sub> , T <sub>1</sub>	Input pins which can be directly tested using conditional branch instructions. T <sub>1</sub> also functions as the event timer input (under software control). T <sub>0</sub> is used during PROM programming and verification in the 8741.
X <sub>1</sub> , X <sub>2</sub>	Inputs for a crystal, LC or an external timing signal to determine the internal oscillator frequency.
SYNC	Output signal which occurs once per UPI-41 instruction cycle. SYNC can be used as a strobe for external circuitry; it is also used to synchronize single step operation.
EA	External access input which allows emulation, testing and PROM/ROM verification.
PROG	Multifunction pin used as the program pulse input during PROM programming. During I/O expander access the PROG pin acts as an address/data strobe to the 8243.
$\overline{RESET}$	Input used to reset status flip-flops and to set the program counter to zero. $\overline{RESET}$ is also used during PROM programming and verification.
$\overline{SS}$	Single step input used in the 8741 in conjunction with the SYNC output to step the program through each instruction.
V <sub>cc</sub>	+5V power supply pin.
V <sub>DD</sub>	+5V during normal operation. Programming supply pin during PROM programming. Low power standby pin in ROM version.
V <sub>ss</sub>	Circuit ground potential.

## UPI INSTRUCTION SET

Mnemonic	Description	Bytes	Cycles
<b>ACCUMULATOR</b>			
ADD A,Rr	Add register to A	1	1
ADD A,@Rr	Add data memory to A	1	1
ADD A,#data	Add immediate to A	2	2
ADDC A,Rr	Add immed. to A with carry	1	1
ADDC A,@Rr	Add immed. to A with carry	1	1
ADDC A,#data	Add immed. to A with carry	2	2
ANL A,Rr	AND register to A	1	1
ANL A,@Rr	AND data memory to A	1	1
ANL A,#data	AND immediate to A	2	2
ORL A,Rr	OR register to A	1	1
ORL A,@Rr	OR data memory to A	1	1
ORL A,#data	OR immediate to A	2	2
XRL A,Rr	Exclusive OR register to A	1	1
XRL A,@Rr	Exclusive OR data memory to A	1	1
XRL A,#data	Exclusive OR immediate to A	2	2
INC A	Increment A	1	1
DEC A	Decrement A	1	1
CLR A	Clear A	1	1
CPL A	Complement A	1	1
DA A	Decimal Adjust A	1	1
SWAP A	Swap digits of A	1	1
RL A	Rotate A left	1	1
RLC A	Rotate A left through carry	1	1
RR A	Rotate A right	1	1
RRC A	Rotate A right through carry	1	1
<b>INPUT/OUTPUT</b>			
IN A,Pp	Input port to A	1	2
OUTL Pp,A	Output A to port	1	2
ANL Pp,#data	AND immediate to port	2	2
ORL Pp,#data	OR immediate to port	2	2
IN A,DBB	Input DBB to A, clear IBF	1	1
OUT DBB,A	Output A to DBB, set OBF	1	1
MOVD A,Pp	Input Expander port to A	1	2
MOVD Pp,A	Output A to Expander port	1	2
ANLD Pp,A	AND A to Expander port	1	2
ORLD Pp,A	OR A to Expander port	1	2
<b>DATA MOVES</b>			
MOV A,Rr	Move register to A	1	1
MOV A,@Rr	Move data memory to A	1	1
MOV A,#data	Move immediate to A	2	2
MOV Rr,A	Move A to register	1	1
MOV @Rr,A	Move A to data memory	1	1
MOV Rr,#data	Move immediate to register	2	2
MOV @Rr,#data	Move immediate to data memory	2	2
MOV A,PSW	Move PSW to A	1	1
MOV PSW,A	Move A to PSW	1	1
XCH A,Rr	Exchange A and register	1	1
XCH A,@Rr	Exchange A and data memory	1	1
XCHD A,@Rr	Exchange digit of A and register	1	1
MOVP A,@A	Move to A from current page	1	2
MOVP3 A,@A	Move to A from page 3	1	2
<b>TIMER/COUNTER</b>			
MOV A,T	Read Timer/Counter	1	1
MOV T,A	Load Timer/Counter	1	1
STRT T	Start Timer	1	1
STRT CNT	Start Counter	1	1
STOP TCNT	Stop Timer/Counter	1	1
EN TCNTI	Enable Timer/Counter Interrupt	1	1
DIS TCNTI	Disable Timer/Counter Interrupt	1	1

Mnemonic	Description	Bytes	Cycles
<b>CONTROL</b>			
EN I	Enable IBF Interrupt	1	1
DIS I	Disable IBF Interrupt	1	1
SEL RB0	Select register bank 0	1	1
SEL RB1	Select register bank 1	1	1
NOP	No Operation	1	1
<b>REGISTERS</b>			
INC Rr	Increment register	1	1
INC @Rr	Increment data memory	1	1
DEC Rr	Decrement register	1	1
<b>SUBROUTINE</b>			
CALL addr	Jump to subroutine	2	2
RET	Return	1	2
RETR	Return and restore status	1	2
<b>FLAGS</b>			
CLR C	Clear Carry	1	1
CPL C	Complement Carry	1	1
CLR F0	Clear Flag 0	1	1
CPL F0	Complement Flag 0	1	1
<b>BRANCH</b>			
JMP addr	Jump unconditional	2	2
JMPP @A	Jump indirect	1	2
DJNZ R,addr	Decrement register and skip	2	2
JC addr	Jump on Carry = 1	2	2
JNC addr	Jump on Carry = 0	2	2
JZ addr	Jump on A Zero	2	2
JNZ addr	Jump on A not Zero	2	2
JT0 addr	Jump on T0 = 1	2	2
JNT0 addr	Jump on T0 = 0	2	2
JT1 addr	Jump on T1 = 1	2	2
JNT1 addr	Jump on T1 = 0	2	2
JF0 addr	Jump on F0 Flag = 1	2	2
JF1 addr	Jump on F1 Flag = 1	2	2
JTF addr	Jump on Timer Flag = 1, Clear Flag	2	2
JNIBF addr	Jump on IBF Flag = 0	2	2
JOBF addr	Jump on OBF Flag = 1	2	2
Jb addr	Jump on Accumulator Bit	2	2

## APPLICATIONS

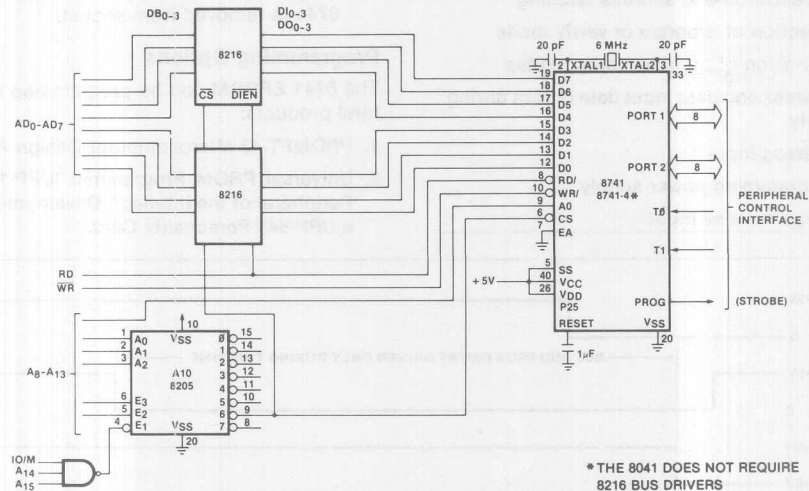


Figure 1. Recommended 8741 Interface to an 8085 System

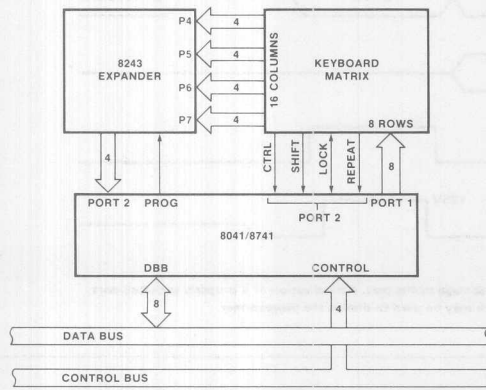


Figure 2. 8041-8243 Keyboard Scanner

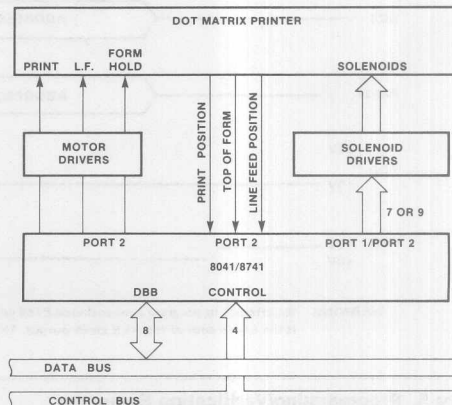


Figure 3. 8041 Matrix Printer Interface

## PROGRAMMING, VERIFYING, AND ERASING THE 8741 EPROM

### Programming/Verification

In brief, the programming process consists of: activating the program mode, applying an address, latching the address, applying data, and applying a programming pulse. Each word is programmed completely before moving on to the next and is followed by a verification step. The following is a list of the pins used for programming and a description of their functions:

Pin	Function
XTAL 1	Clock input (1 to 6 MHz)
RESET	Initialization and address latching
TEST 0	Selection of program or verify mode
EA	Activation of program/verify modes
BUS	Address and data input data output during verify
P20-1	Address input
V <sub>DD</sub>	Programming power supply
PROG	Program pulse input

The program/verify sequence is:

1. V<sub>DD</sub> = 5V, clock applied or internal oscillator operating, RESET = 0V, TEST 0 = 5V, EA = 5V, BUS and PROG floating.
2. Insert 8741 in programming socket.
3. TEST 0 = 0V (select program mode).
4. EA = 25V (activate program mode).
5. Address applied to BUS and P20-1.
6. RESET = 5V (latch address).
7. Data applied to BUS.
8. V<sub>D</sub> = 25V (programming power).
9. PROG = 0V followed by one 50 ms pulse to 25V.
10. V<sub>DD</sub> = 5V.
11. TEST 0 = 5V (verify mode).
12. Read and verify data on BUS.
13. TEST 0 = 0V.
14. RESET = 0V and repeat from step 5.
15. Programmer should be at conditions of step 1 when 8741 is removed from socket.

### Programming Options

The 8741 EPROM can be programmed by either of two Intel products:

1. PROMPT-48 Microcomputer Design Aid.
2. Universal PROM Programmer (UPP-101 or UPP-102) Peripheral of the Intellec® Development System with a UPP-848 Personality Card.

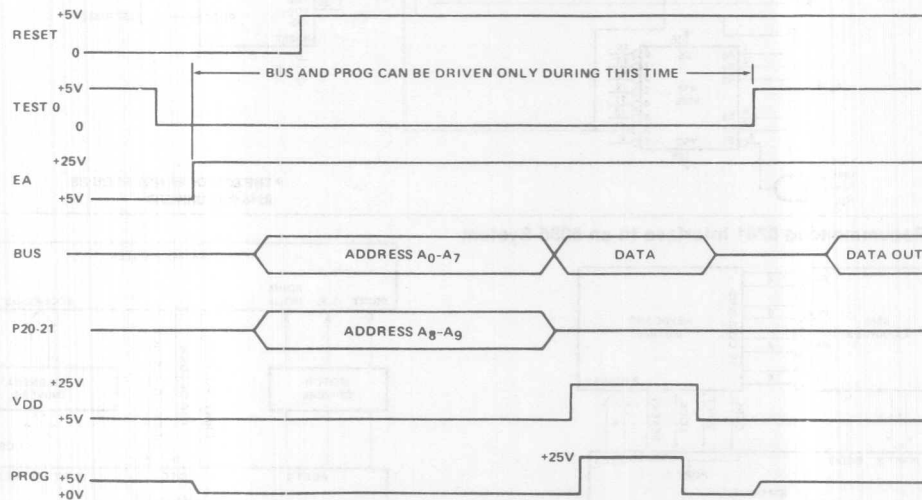


Figure 5. Programming/Verification Sequence

### 8741 Erasure Characteristics

The erasure characteristics of the 8741 are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000–4000 Å range. Data show that constant exposure to room level fluorescent lighting could erase the typical 8748 in approximately 3 years while it would take approximately one week to cause erasure when exposed to direct sunlight. If the 8741 is to be exposed to these types of lighting conditions for extended periods of

time, opaque labels are available from Intel which should be placed over the 8741 window to prevent unintentional erasure.

The recommended erasure procedure for the 8741 is exposure to shortwave ultraviolet light which has a wavelength of 2537 Å. The integrated dose (i.e., UV intensity  $\times$  exposure time) for erasure should be a minimum of 15 W-sec/cm<sup>2</sup>. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12,000  $\mu$ W/cm<sup>2</sup> power rating. The 8741 should be placed within one inch of the lamp tubes during erasure. Some lamps have a filter on their tubes which should be removed before erasure.

### A.C. TIMING SPECIFICATION FOR PROGRAMMING

$T_A = 25^\circ\text{C} \pm 5^\circ\text{C}$ ,  $V_{CC} = 5\text{V} \pm 5\%$ ,  $V_{DD} = 25\text{V} \pm 1\text{V}$

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t <sub>AW</sub>	Address Setup Time to $\overline{\text{RESET}}$ 1	4tcy			
t <sub>WA</sub>	Address Hold Time After $\overline{\text{RESET}}$ 1	4tcy			
t <sub>DW</sub>	Data in Setup Time to PROG 1	4tcy			
t <sub>WD</sub>	Data in Hold Time After PROG 1	4tcy			
t <sub>PH</sub>	$\overline{\text{RESET}}$ Hold Time to Verify	4tcy			
t <sub>VDDW</sub>	V <sub>DD</sub>	4tcy			
t <sub>VDDH</sub>	V <sub>DD</sub> Hold Time After PROG 1	0			
t <sub>PW</sub>	Program Pulse Width	50	60	MS	
t <sub>TW</sub>	Test 0 Setup Time for Program Mode	4tcy			
t <sub>WT</sub>	Test 0 Hold Time After Program Mode	4tcy			
t <sub>DO</sub>	Test 0 to Data Out Delay		4tcy		
t <sub>WW</sub>	$\overline{\text{RESET}}$ Pulse Width to Latch Address	4tcy			
t <sub>r</sub> , t <sub>f</sub>	V <sub>DD</sub> and PROG Rise and Fall Times	0.5	2.0	$\mu$ S	
t <sub>CY</sub>	CPU Operation Cycle Time	5.0		$\mu$ S	
t <sub>RE</sub>	$\overline{\text{RESET}}$ Setup Time Before EA 1	4tcy			

Note: If TEST 0 is high, t<sub>DO</sub> can be triggered by  $\overline{\text{RESET}}$  1.

### D.C. SPECIFICATION FOR PROGRAMMING

$T_A = 25^\circ\text{C} \pm 5^\circ\text{C}$ ,  $V_{CC} = 5\text{V} \pm 5\%$ ,  $V_{DD} = 25\text{V} \pm 1\text{V}$

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V <sub>DOH</sub>	V <sub>DD</sub> Program Voltage High Level	24.0	26.0	V	
V <sub>DDL</sub>	V <sub>DD</sub> Voltage Low Level	4.75	5.25	V	
V <sub>PH</sub>	PROG Program Voltage High Level	21.5	24.5	V	
V <sub>PL</sub>	PROG Voltage Low Level		0.2	V	
V <sub>EAH</sub>	EA Program or Verify Voltage High Level	21.5	24.5	V	
V <sub>EAL</sub>	EA Voltage Low Level		5.25	V	
I <sub>DD</sub>	V <sub>DD</sub> High Voltage Supply Current		30.0	mA	
I <sub>PROG</sub>	PROG High Voltage Supply Current		16.0	mA	
I <sub>EA</sub>	EA High Voltage Supply Current		1.0	mA	

# 8205

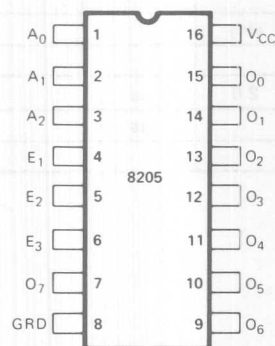
## HIGH SPEED 1 OUT OF 8 BINARY DECODER

- I/O Port or Memory Selector
- Simple Expansion — Enable Inputs
- High Speed Schottky Bipolar Technology — 18 ns Max Delay
- Directly Compatible with TTL Logic Circuits
- Low Input Load Current — 0.25 mA Max, 1/6 Standard TTL Input Load
- Minimum Line Reflection — Low Voltage Diode Input Clamp
- Outputs Sink 10 mA Min
- 16-Pin Dual In-Line Ceramic or Plastic Package

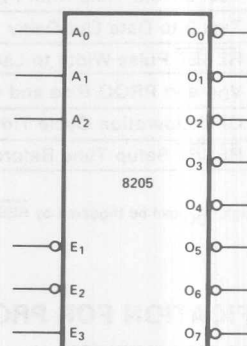
The Intel® 8205 decoder can be used for expansion of systems which utilize input ports, output ports, and memory components with active low chip select input. When the 8205 is enabled, one of its 8 outputs goes "low", thus a single row of a memory system is selected. The 3-chip enable inputs on the 8205 allow easy system expansion. For very large systems, 8205 decoders can be cascaded such that each decoder can drive 8 other decoders for arbitrary memory expansions.

The 8205 is packaged in a standard 16-pin dual in-line package, and its performance is specified over the temperature range of 0°C to +75°C, ambient. The use of Schottky barrier diode clamped transistors to obtain fast switching speeds results in higher performance than equivalent devices made with a gold diffusion process.

PIN CONFIGURATION



LOGIC SYMBOL



PIN NAMES

A <sub>0</sub> , A <sub>2</sub>	ADDRESS INPUTS
E <sub>1</sub> , E <sub>3</sub>	ENABLE INPUTS
O <sub>0</sub> , O <sub>7</sub>	DECODED OUTPUTS

ADDRESS			ENABLE			OUTPUTS							
A <sub>0</sub>	A <sub>1</sub>	A <sub>2</sub>	E <sub>1</sub>	E <sub>2</sub>	E <sub>3</sub>	0	1	2	3	4	5	6	7
L	L	L	L	L	H	L	H	H	H	H	H	H	H
H	L	L	L	L	H	H	L	H	H	H	H	H	H
H	H	L	L	L	H	H	H	L	H	H	H	H	H
L	L	H	L	L	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	H	L	H	H	H
L	H	H	L	L	H	H	H	H	H	H	L	H	H
H	H	H	L	L	H	H	H	H	H	H	H	L	H
X	X	X	L	L	L	H	H	H	H	H	H	H	H
X	X	X	H	L	L	H	H	H	H	H	H	H	H
X	X	X	L	L	H	H	H	H	H	H	H	H	H
X	X	X	H	H	L	H	H	H	H	H	H	H	H
X	X	X	H	L	H	H	H	H	H	H	H	H	H
X	X	X	L	H	H	H	H	H	H	H	H	H	H
X	X	X	H	H	H	H	H	H	H	H	H	H	H

## FUNCTIONAL DESCRIPTION

### Decoder

The 8205 contains a one out of eight binary decoder. It accepts a three bit binary code and by gating this input, creates an exclusive output that represents the value of the input code.

For example, if a binary code of 101 was present on the A0, A1 and A2 address input lines, and the device was enabled, an active low signal would appear on the  $\overline{O_5}$  output line. Note that all of the other output pins are sitting at a logic high, thus the decoded output is said to be exclusive. The decoders outputs will follow the truth table shown below in the same manner for all other input variations.

### Enable Gate

When using a decoder it is often necessary to gate the outputs with timing or enabling signals so that the exclusive output of the decoded value is synchronous with the overall system.

The 8205 has a built-in function for such gating. The three enable inputs ( $\overline{E_1}$ ,  $\overline{E_2}$ , E3) are ANDed together and create a single enable signal for the decoder. The combination of both active "high" and active "low" device enable inputs provides the designer with a powerfully flexible gating function to help reduce package count in his system.

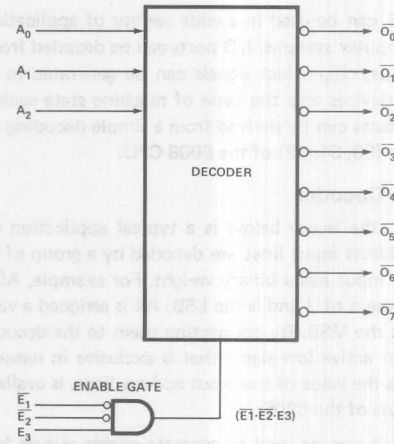


Figure 1. Enable Gate

ADDRESS			ENABLE			OUTPUTS							
A <sub>0</sub>	A <sub>1</sub>	A <sub>2</sub>	E <sub>1</sub>	E <sub>2</sub>	E <sub>3</sub>	0	1	2	3	4	5	6	7
L	L	L	L	L	H	L	H	H	H	H	H	H	H
H	L	L	L	L	H	H	L	H	H	H	H	H	H
L	H	L	L	L	H	H	H	L	H	H	H	H	H
H	H	L	L	L	H	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
H	L	H	L	L	H	H	H	H	H	H	L	H	H
L	H	H	L	L	H	H	H	H	H	H	H	L	H
H	H	H	L	L	H	H	H	H	H	H	H	H	L
X	X	X	L	L	L	H	H	H	H	H	H	H	H
X	X	X	H	L	L	H	H	H	H	H	H	H	H
X	X	X	L	H	L	H	H	H	H	H	H	H	H
X	X	X	H	H	L	H	H	H	H	H	H	H	H
X	X	X	H	L	H	H	H	H	H	H	H	H	H
X	X	X	L	H	H	H	H	H	H	H	H	H	H
X	X	X	H	H	H	H	H	H	H	H	H	H	H

## APPLICATIONS OF THE 8205

The 8205 can be used in a wide variety of applications in microcomputer systems. I/O ports can be decoded from the address bus, chip select signals can be generated to select memory devices and the type of machine state such as in 8008 systems can be derived from a simple decoding of the state lines (S0, S1, S2) of the 8008 CPU.

### I/O Port Decoder

Shown in the figure below is a typical application of the 8205. Address input lines are decoded by a group of 8205s (3). Each input has a binary weight. For example, A0 is assigned a value of 1 and is the LSB; A4 is assigned a value of 16 and is the MSB. By connecting them to the decoders as shown, an active low signal that is exclusive in nature and represents the value of the input address lines, is available at the outputs of the 8205s.

This circuit can be used to generate enable signals for I/O ports or any other decoder related application.

Note that no external gating is required to decode up to 24 exclusive devices and that a simple addition of an inverter or two will allow expansion to even larger decoder networks.

### Chip Select Decoder

Using a very similar circuit to the I/O port decoder, an ar-

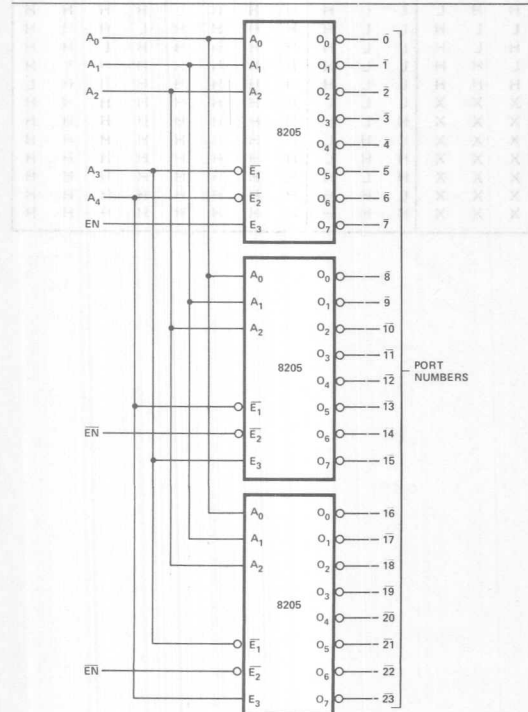


Figure 2. I/O Port Decoder

ray of 8205s can be used to create a simple interface to a 24K memory system.

The memory devices used can be either ROM or RAM and are 1K in storage capacity. 8308s and 8102s are the devices typically used for this application. This type of memory device has ten (10) address inputs and an active "low" chip select (CS). The lower order address bits A0-A9 which come from the microprocessor are "bussed" to all memory elements and the chip select to enable a specific device or group of devices comes from the array of 8205s. The output of the 8205 is active low so it is directly compatible with the memory components.

Basic operation is that the CPU issues an address to identify a specific memory location in which it wishes to "write" or "read" data. The most significant address bits A10-A14 are decoded by the array of 8205s and an exclusive, active low, chip select is generated that enables a specific memory device. The least significant address bits A0-A9 identify a specific location within the selected device. Thus, all addresses throughout the entire memory array are exclusive in nature and are non-redundant.

This technique can be expanded almost indefinitely to support even larger systems with the addition of a few inverters and an extra decoder (8205).

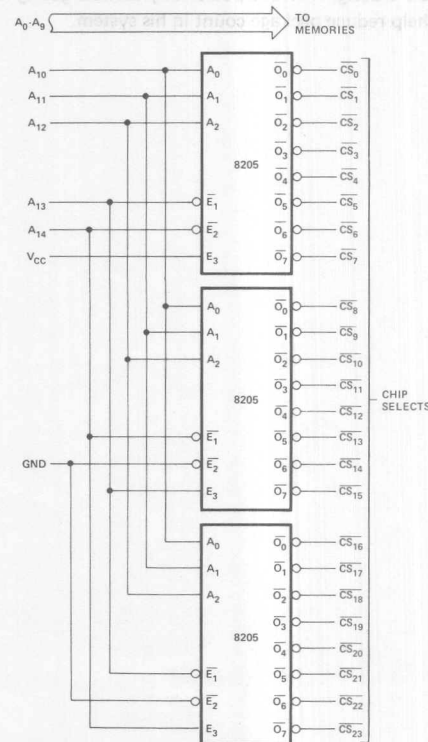


Figure 3. 32K Memory Interface

### Logic Element Example

Probably the most overlooked application of the 8205 is that of a general purpose logic element. Using the "on-chip" enabling gate, the 8205 can be configured to gate its decoded outputs with system timing signals and generate strobes that can be directly connected to latches, flip-flops and one-shots that are used throughout the system.

An excellent example of such an application is the "state decoder" in an 8008 CPU based system. The 8008 CPU issues three bits of information (S<sub>0</sub>, S<sub>1</sub>, S<sub>2</sub>) that indicate the nature of the data on the Data Bus during each machine state. Decoding of these signals is vital to generate strobes that can load the address latches, control bus discipline and general machine functions.

In the figure below a circuit is shown using the 8205 as the "state decoder" for an 8008 CPU that not only decodes the S<sub>0</sub>, S<sub>1</sub>, S<sub>2</sub> outputs but gates these signals with the clock (phase 2) and the SYNC output of the 8008 CPU. The  $\overline{T1}$

and  $\overline{T2}$  decoded strobes can connect directly to devices like 8212s for latching the address information. The other decoded strobes can be used to generate signals to control the system data bus, memory timing functions and interrupt structure. RESET is connected to the enable gate so that strobes are not generated during system reset, eliminating accidental loading.

The power of such a circuit becomes evident when a single decoded strobe is logically broken down. Consider  $\overline{T1}$  output, the boolean equation for it would be:

$$\overline{T1} = (\overline{S0} \cdot S1 \cdot S2) \cdot (\overline{SYNC} \cdot \text{Phase 2} \cdot \overline{\text{Reset}})$$

A six input NAND gate plus a few inverters would be needed to implement this function. The seven remaining outputs would need a similar circuit to duplicate their function, obviously a substantial savings in components can be achieved when using such a technique.

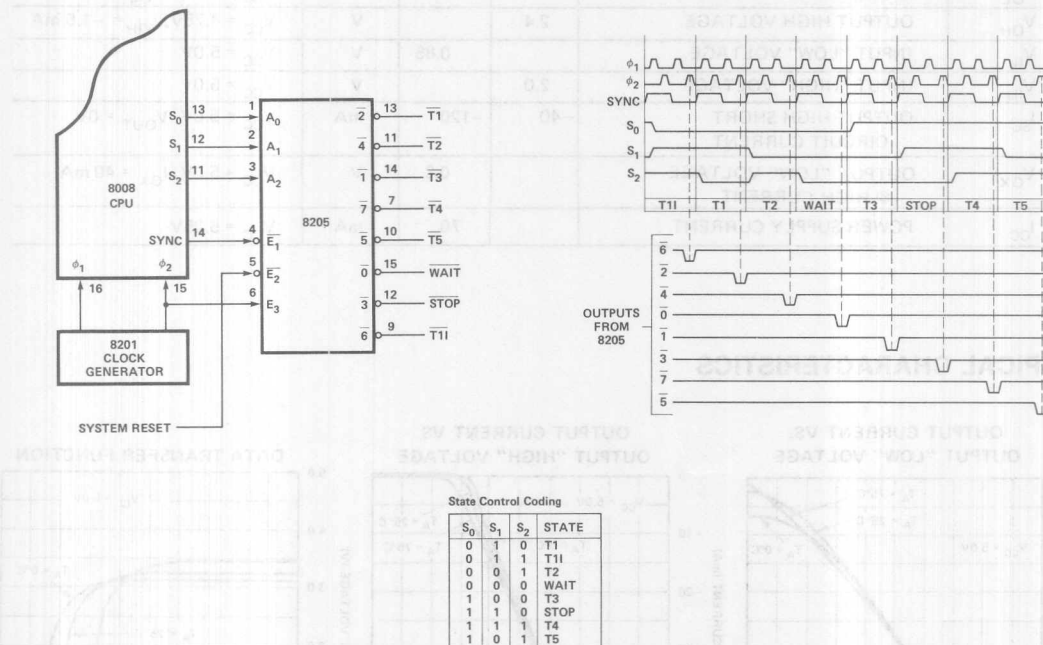


Figure 4. 8205 State Decoder Circuit

Temperature Under Bias:	Ceramic	-65°C to +125°C
	Plastic	-65°C to +75°C
Storage Temperature		-65°C to +160°C
All Output or Supply Voltages		-0.5 to +7 Volts
All Input Voltages		-1.0 to +5.5 Volts
Output Currents		125 mA

#### \*COMMENT

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

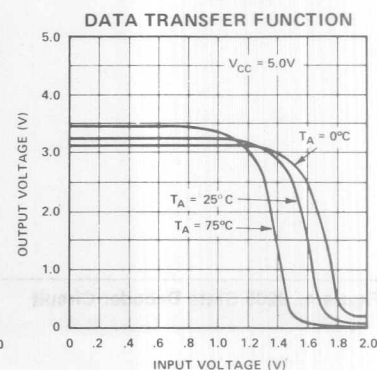
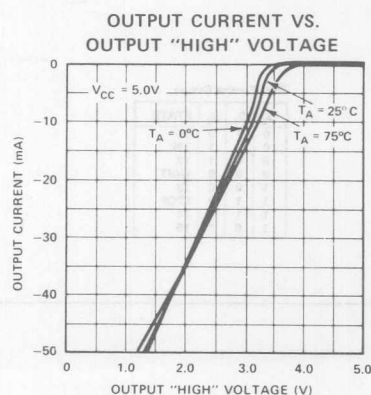
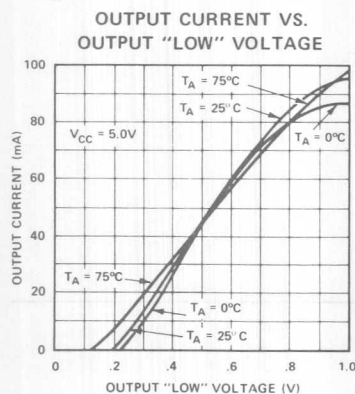
## D.C. CHARACTERISTICS

$T_A = 0^\circ\text{C to } +75^\circ\text{C}$ ,  $V_{CC} = 5\text{V} \pm 5\%$

8205

SYMBOL	PARAMETER	LIMIT		UNIT	TEST CONDITIONS
		MIN.	MAX.		
$I_F$	INPUT LOAD CURRENT		-0.25	mA	$V_{CC} = 5.25\text{V}$ , $V_F = 0.45\text{V}$
$I_R$	INPUT LEAKAGE CURRENT		10	$\mu\text{A}$	$V_{CC} = 5.25\text{V}$ , $V_R = 5.25\text{V}$
$V_C$	INPUT FORWARD CLAMP VOLTAGE		-1.0	V	$V_{CC} = 4.75\text{V}$ , $I_C = -5.0\text{ mA}$
$V_{OL}$	OUTPUT "LOW" VOLTAGE		0.45	V	$V_{CC} = 4.75\text{V}$ , $I_{OL} = 10.0\text{ mA}$
$V_{OH}$	OUTPUT HIGH VOLTAGE	2.4		V	$V_{CC} = 4.75\text{V}$ , $I_{OH} = -1.5\text{ mA}$
$V_{IL}$	INPUT "LOW" VOLTAGE		0.85	V	$V_{CC} = 5.0\text{V}$
$V_{IH}$	INPUT "HIGH" VOLTAGE	2.0		V	$V_{CC} = 5.0\text{V}$
$I_{SC}$	OUTPUT HIGH SHORT CIRCUIT CURRENT	-40	-120	mA	$V_{CC} = 5.0\text{V}$ , $V_{OUT} = 0\text{V}$
$V_{OX}$	OUTPUT "LOW" VOLTAGE @ HIGH CURRENT		0.8	V	$V_{CC} = 5.0\text{V}$ , $I_{OX} = 40\text{ mA}$
$I_{CC}$	POWER SUPPLY CURRENT		70	mA	$V_{CC} = 5.25\text{V}$

## TYPICAL CHARACTERISTICS



## SWITCHING CHARACTERISTICS

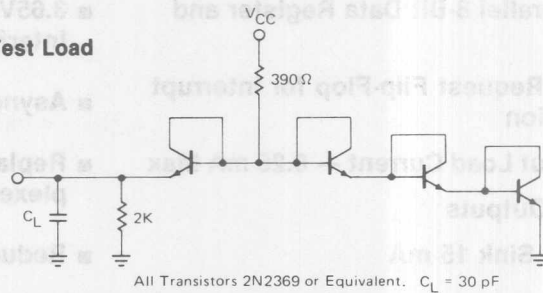
### Conditions of Test:

Input pulse amplitudes: 2.5V

Input rise and fall times: 5 nsec  
between 1V and 2V

Measurements are made at 1.5V

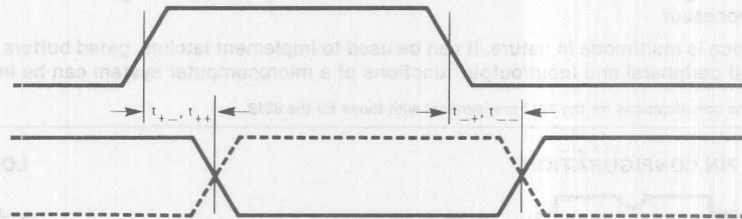
### Test Load



### Test Waveforms

ADDRESS OR ENABLE  
INPUT PULSE

OUTPUT



## A.C. CHARACTERISTICS

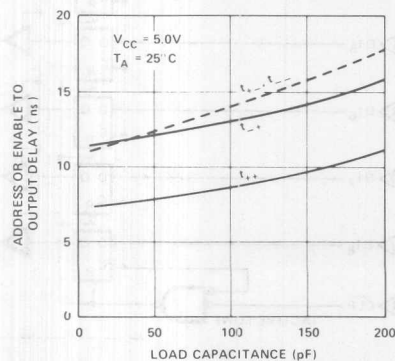
$T_A = 0^\circ\text{C to } +75^\circ\text{C}$ ,  $V_{CC} = 5V \pm 5\%$  unless otherwise specified.

SYMBOL	PARAMETER	MAX. LIMIT	UNIT	TEST CONDITIONS
$t_{++}$	ADDRESS OR ENABLE TO OUTPUT DELAY	18	ns	$f = 1 \text{ MHz}$ , $V_{CC} = 0V$ $V_{BIAS} = 2.0V$ , $T_A = 25^\circ\text{C}$
$t_{-+}$		18	ns	
$t_{+-}$		18	ns	
$t_{--}$		18	ns	
$C_{IN}^{(1)}$	INPUT CAPACITANCE	4(typ.)	pF	$f = 1 \text{ MHz}$ , $V_{CC} = 0V$ $V_{BIAS} = 2.0V$ , $T_A = 25^\circ\text{C}$
	P8205 C8205	5(typ.)	pF	

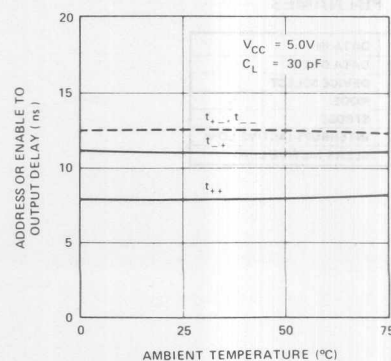
1. This parameter is periodically sampled and is not 100% tested.

## TYPICAL CHARACTERISTICS

ADDRESS OR ENABLE TO OUTPUT  
DELAY VS. LOAD CAPACITANCE



ADDRESS OR ENABLE TO OUTPUT  
DELAY VS. AMBIENT TEMPERATURE



# 8-BIT INPUT/OUTPUT PORT

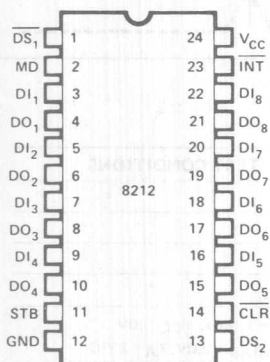
- Fully Parallel 8-Bit Data Register and Buffer
- Service Request Flip-Flop for Interrupt Generation
- Low Input Load Current — 0.25 mA Max
- 3-State Outputs
- Outputs Sink 15 mA
- 3.65V Output High Voltage for Direct Interface to 8080 CPU or 8008 CPU
- Asynchronous Register Clear
- Replaces Buffers, Latches, and Multiplexers in Microcomputer Systems
- Reduces System Package Count

The Intel® 8212 input/output port consists of an 8-bit latch with 3-state output buffers along with control and device selection logic. Also included is a service request flip-flop for the generation and control of interrupts to the microprocessor.

The device is multimode in nature. It can be used to implement latches, gated buffers or multiplexers. Thus, all of the principal peripheral and input/output functions of a microcomputer system can be implemented with this device.

\*Note: The specifications for the 3212 are identical with those for the 8212.

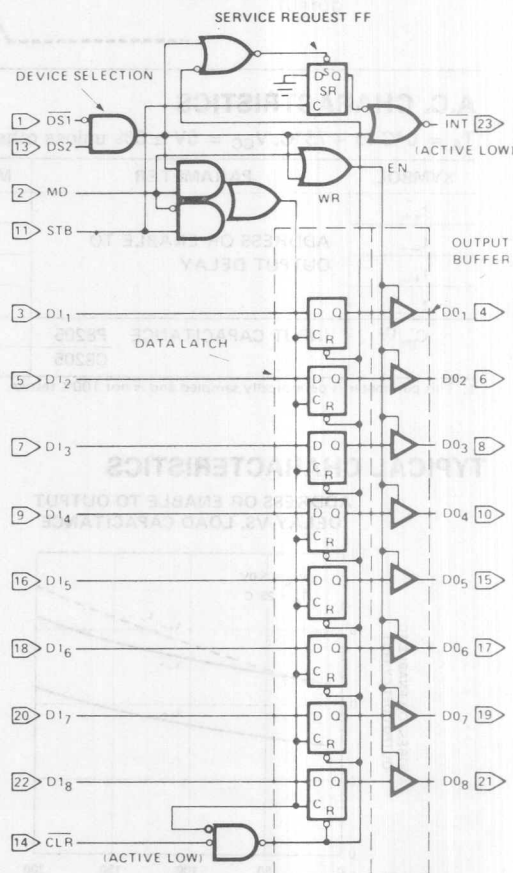
## PIN CONFIGURATION



## PIN NAMES

DI <sub>1</sub> -DI <sub>8</sub>	DATA IN
DO <sub>1</sub> -DO <sub>8</sub>	DATA OUT
DS <sub>1</sub> -DS <sub>2</sub>	DEVICE SELECT
MD	MODE
STB	STROBE
INT	INTERRUPT (ACTIVE LOW)
CLR	CLEAR (ACTIVE LOW)

## LOGIC DIAGRAM



## FUNCTIONAL DESCRIPTION

### Data Latch

The 8 flip-flops that make up the data latch are of a "D" type design. The output (Q) of the flip-flop will follow the data input (D) while the clock input (C) is high. Latching will occur when the clock (C) returns low.

The data latch is cleared by an asynchronous reset input ( $\overline{\text{CLR}}$ ). (Note: Clock (C) Overrides Reset ( $\overline{\text{CLR}}$ ).)

### Output Buffer

The outputs of the data latch (Q) are connected to 3-state, non-inverting output buffers. These buffers have a common control line (EN); this control line either enables the buffer to transmit the data from the outputs of the data latch (Q) or disables the buffer, forcing the output into a high impedance state. (3-state)

This high-impedance state allows the designer to connect the 8212 directly onto the microprocessor bi-directional data bus.

### Control Logic

The 8212 has control inputs  $\overline{\text{DS1}}$ , DS2, MD and STB. These inputs are used to control device selection, data latching, output buffer state and service request flip-flop.

### DS1, DS2 (Device Select)

These 2 inputs are used for device selection. When  $\overline{\text{DS1}}$  is low and DS2 is high ( $\overline{\text{DS1}} \cdot \text{DS2}$ ) the device is selected. In the selected state the output buffer is enabled and the service request flip-flop (SR) is asynchronously set.

### MD (Mode)

This input is used to control the state of the output buffer and to determine the source of the clock input (C) to the data latch.

When MD is high (output mode) the output buffers are enabled and the source of clock (C) to the data latch is from the device selection logic ( $\overline{\text{DS1}} \cdot \text{DS2}$ ). When MD is low (input mode) the output buffer state is determined by the device selection logic ( $\overline{\text{DS1}} \cdot \text{DS2}$ ) and the source of clock (C) to the data latch is the STB (Strobe) input.

### STB (Strobe)

This input is used as the clock (C) to the data latch for the input mode MD = 0 and to synchronously reset the service request flip-flop (SR).

Note that the SR flip-flop is negative edge triggered.

### Service Request Flip-Flop

The (SR) flip-flop is used to generate and control interrupts in microcomputer systems. It is asynchronously set by the CLR input (active low). When the (SR) flip-flop is set it is in the non-interrupting state.

The output of the (SR) flip-flop (Q) is connected to an inverting input of a "NOR" gate. The other input to the "NOR" gate is non-inverting and is connected to the device selection logic ( $\overline{\text{DS1}} \cdot \text{DS2}$ ). The output of the "NOR" gate (INT) is active low (interrupting state) for connection to active low input priority generating circuits.

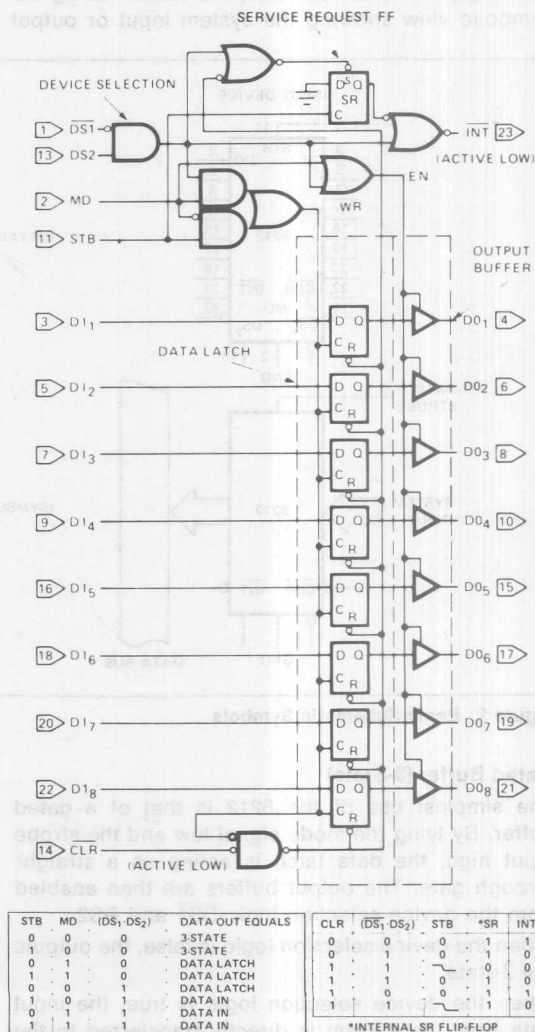


Figure 1. Service Flip-Flop Function

## APPLICATIONS OF THE 8212 — FOR MICROCOMPUTER SYSTEMS

- Basic schematic symbols
- Gated buffer
- Bidirectional bus driver
- Interrupting input port

### Basic Schematic Symbols

Two examples of ways to draw the 8212 on system schematics—(1) the top being the detailed view showing pin numbers, and (2) the bottom being the symbolic view showing the system input or output

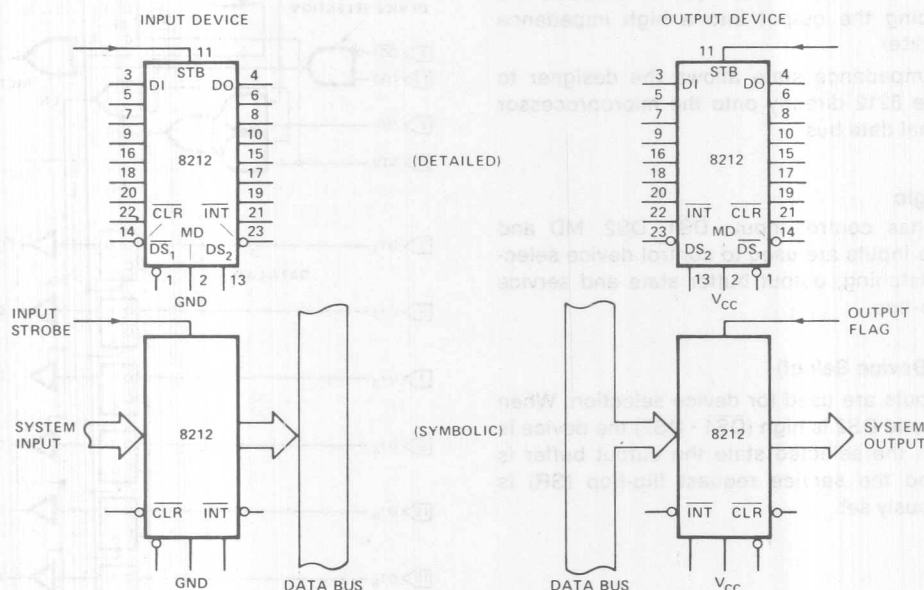


Figure 2. Basic Schematic Symbols

### Gated Buffer (3-State)

The simplest use of the 8212 is that of a gated buffer. By tying the mode signal low and the strobe input high, the data latch is acting as a straight through gate. The output buffers are then enabled from the device selection logic  $\overline{DS1}$  and  $DS2$ .

When the device selection logic is false, the outputs are 3-state.

When the device selection logic is true, the input data from the system is directly transferred to the output. The input data load is 250 micro amps. The output data can sink 15 milli amps. The minimum high output is 3.65 volts.

- Interrupt instruction port
- Output port
- 8080A status latch
- 8085A address latch

as a system bus (bus containing 8 parallel lines). The output to the data bus is symbolic in referencing 8 parallel lines.

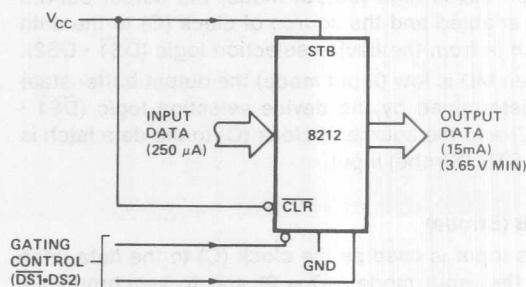


Figure 3. Gated Buffer (3-State)

### Bidirectional Bus Driver

A pair of 8212's wired (back-to-back) can be used as a symmetrical drive, bi-directional bus driver. The devices are controlled by the data bus input control which is connected to  $\overline{DS1}$  on the first 8212 and to DS2 on the second. One device is active, and acting as a straight through buffer the other is in 3-state mode. This is a very useful circuit in small system design.

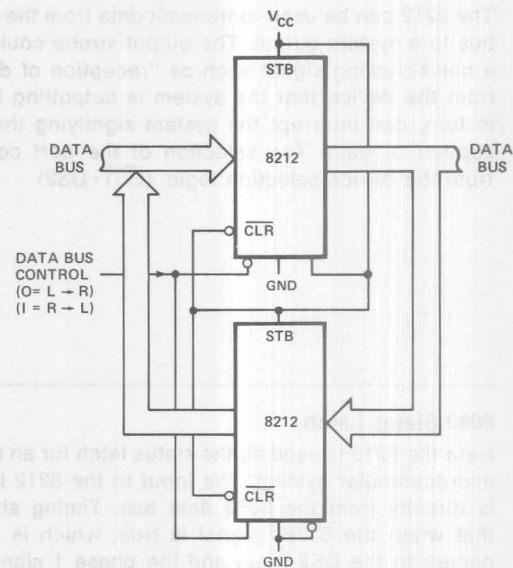


Figure 4. Bidirectional Bus Driver

### Interrupting Input Port

This use of an 8212 is that of a system input port that accepts a strobe from the system input source, which in turn clears the service request flip-flop and interrupts the processor. The processor then goes through a service routine, identifies the port, and causes the device selection logic to go true — enabling the system input data onto the data bus.

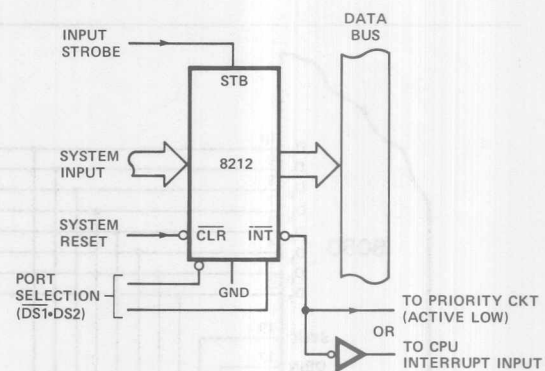


Figure 5. Interrupting Input Port

### Interrupt Instruction Port

The 8212 can be used to gate the interrupt instruction, normally RESTART instructions, onto the data bus. The device is enabled from the interrupt acknowledge signal from the microprocessor and from a port selection signal. This signal is normally tied to ground. ( $\overline{DS1}$  could be used to multiplex a variety of interrupt instruction ports onto a common bus).

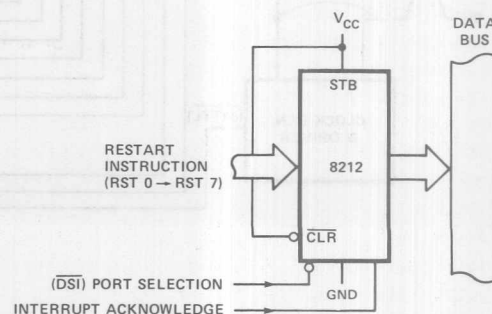


Figure 6. Interrupt Instruction Port

### Output Port (With Handshaking)

The 8212 can be used to transmit data from the data bus to a system output. The output strobe could be a hand-shaking signal such as "reception of data" from the device that the system is outputting to. It in turn, can interrupt the system signifying the reception of data. The selection of the port comes from the device selection logic. ( $\overline{DS1} \cdot DS2$ )

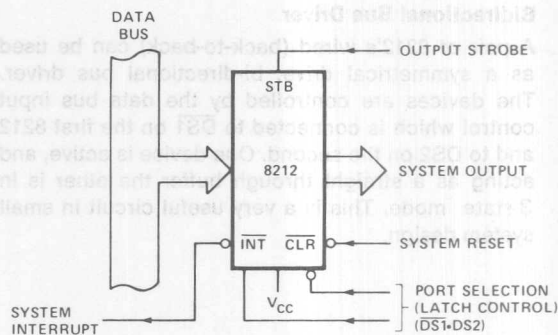


Figure 7. Output Port (With Handshaking)

### 8080 Status Latch

Here the 8212 is used as the status latch for an 8080 microcomputer system. The input to the 8212 latch is directly from the 8080 data bus. Timing shows that when the SYNC signal is true, which is connected to the DS2 input and the phase 1 signal is true, which is a TTL level coming from the clock generator; then, the status data will be latched into the 8212.

Note: The mode signal is tied high so that the output on the latch is active and enabled all the time.

It is shown that the two areas of concern are the bidirectional data bus of the microprocessor and the control bus.

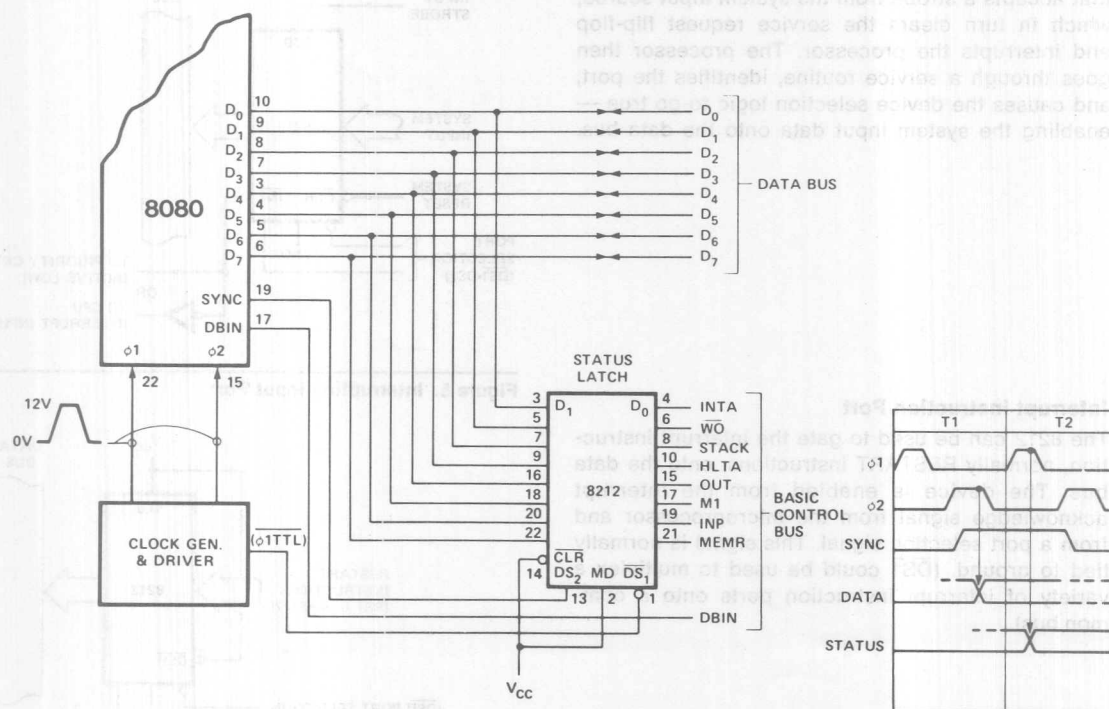


Figure 8. 8080 Status Latch

### 8085A Low-Order Address Latch

The 8085A microprocessor uses a multiplexed address/data bus that contains the low order 8-bits of address information during the first part of a machine cycle. The same bus contains data at a later time in the cycle. An address latch enable (ALE) signal is provided by the 8085A to be used by the 8212 to latch the address so that it may be available through the whole machine cycle. Note: In this configuration, the MODE input is tied high, keeping the 8212's output buffers turned on at all times.

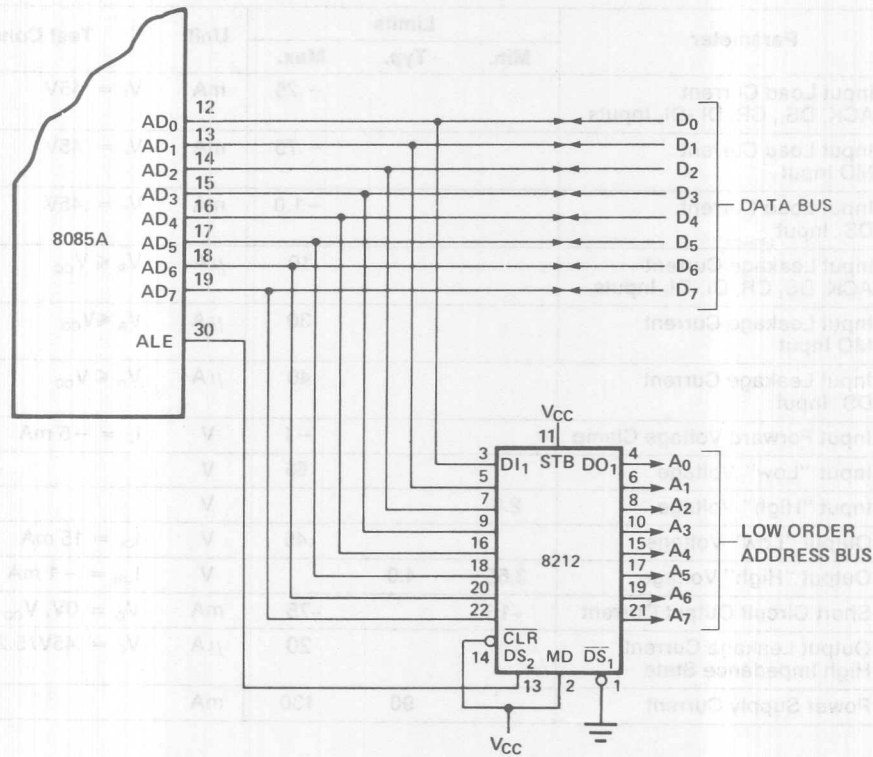


Figure 9. 8085A Low-Order Address Latch

**ABSOLUTE MAXIMUM RATINGS\***

Temperature under bias plastic.....0°C to 75°C  
 Storage temperature.....0°C to 75°C  
 All output or supply voltages.....-0.5V to +7V  
 All input voltages.....-1.0V to +5.5V  
 Output currents.....100 mA

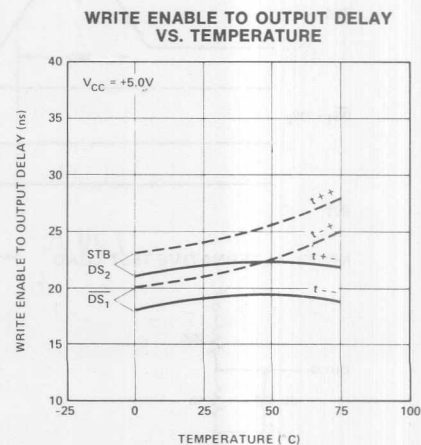
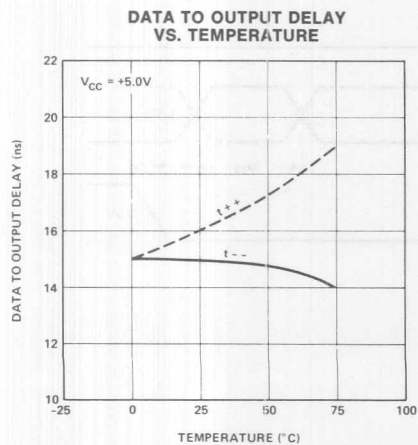
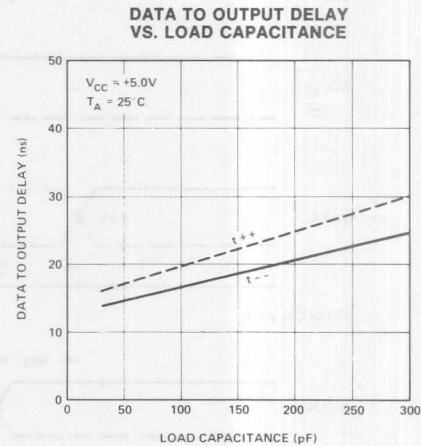
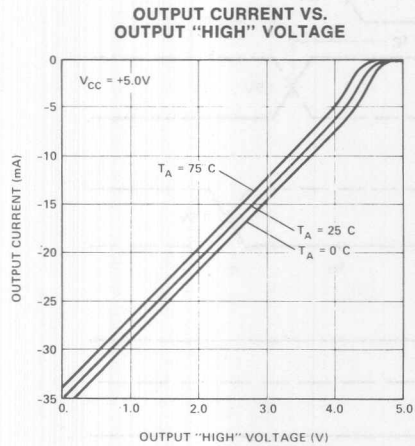
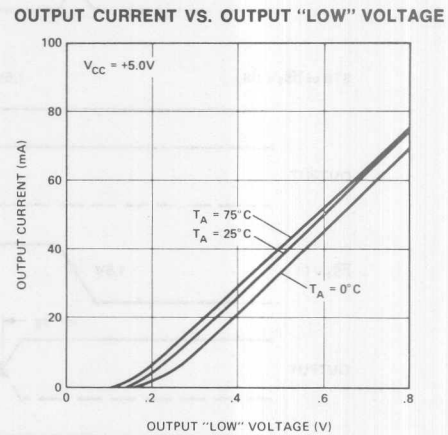
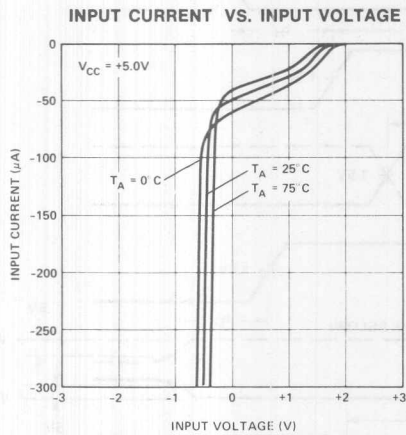
\*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operational sections of this specification is not implied.

**D.C. CHARACTERISTICS**

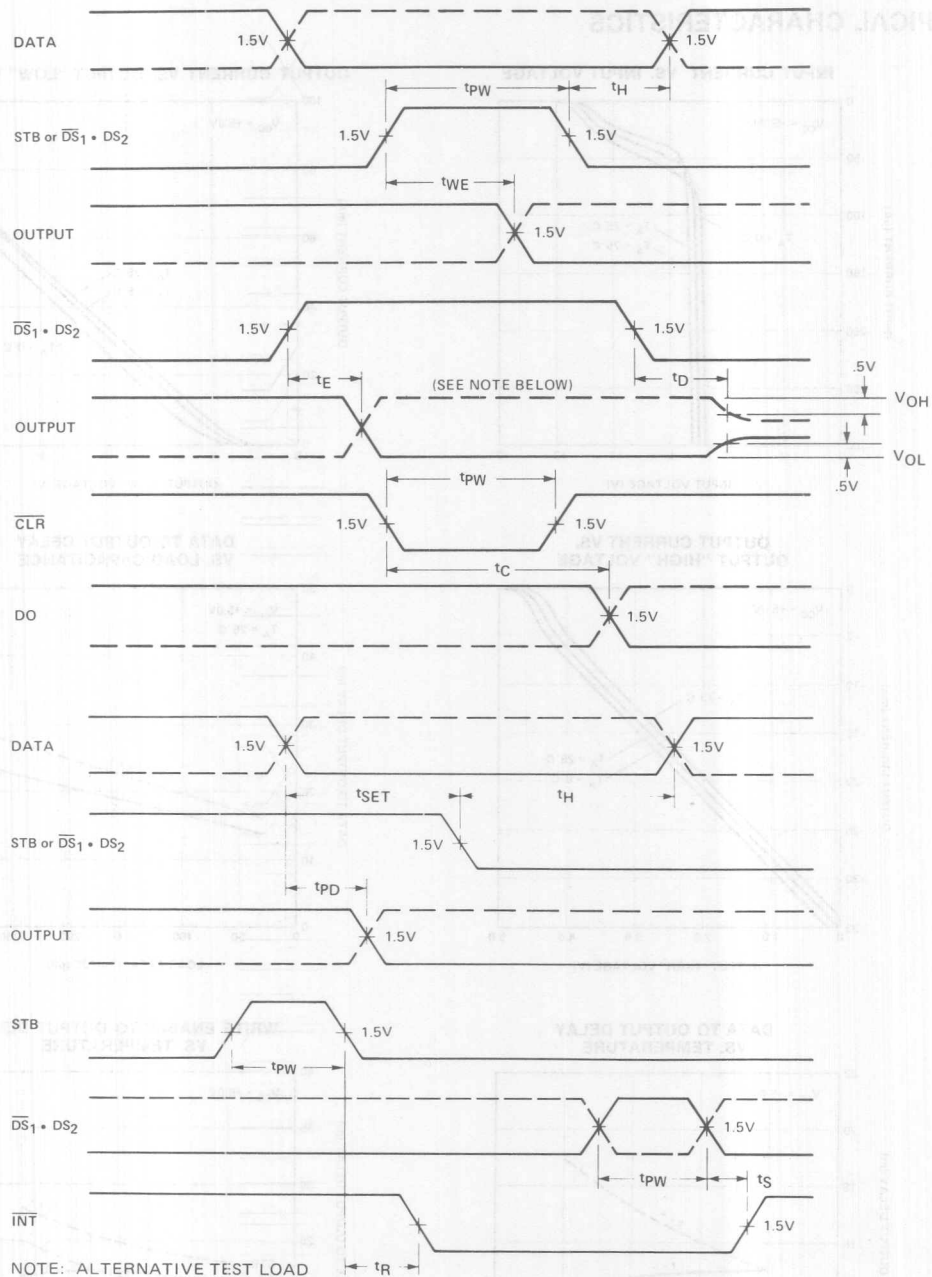
$T_A = 0^\circ\text{C to } +75^\circ\text{C}$   $V_{CC} = +5V \pm 5\%$

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
$I_F$	Input Load Current ACK, DS <sub>2</sub> , CR, DI <sub>1</sub> -DI <sub>8</sub> Inputs			- .25	mA	$V_F = .45V$
$I_F$	Input Load Current MD Input			- .75	mA	$V_F = .45V$
$I_F$	Input Load Current DS <sub>1</sub> Input			- 1.0	mA	$V_F = .45V$
$I_R$	Input Leakage Current ACK, DS, CR, DI <sub>1</sub> -DI <sub>8</sub> Inputs			10	$\mu A$	$V_R \leq V_{CC}$
$I_R$	Input Leakage Current MO Input			30	$\mu A$	$V_R \leq V_{CC}$
$I_R$	Input Leakage Current DS Input			40	$\mu A$	$V_R \leq V_{CC}$
$V_C$	Input Forward Voltage Clamp			- 1	V	$I_C = -5 \text{ mA}$
$V_{IL}$	Input "Low" Voltage			.85	V	
$V_{IH}$	Input "High" Voltage	2.0			V	
$V_{OL}$	Output "Low" Voltage			.45	V	$I_{OL} = 15 \text{ mA}$
$V_{OH}$	Output "High" Voltage	3.65	4.0		V	$I_{OH} = -1 \text{ mA}$
$I_{SC}$	Short Circuit Output Current	-15		-75	mA	$V_O = 0V, V_{CC} = 5.0V$
$I_O$	Output Leakage Current High Impedance State			20	$\mu A$	$V_O = .45V/5.25V$
$I_{CC}$	Power Supply Current		90	130	mA	

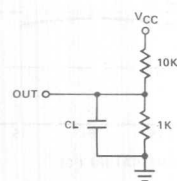
## TYPICAL CHARACTERISTICS



## TIMING DIAGRAM



NOTE: ALTERNATIVE TEST LOAD



**A.C. CHARACTERISTICS**
 $T_A = 0^\circ\text{C to } +75^\circ\text{C}, V_{CC} = +5\text{V} \pm 5\%$ 

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
$t_{pw}$	Pulse Width	25			ns	
$t_{pd}$	Data To Output Delay			30	ns	
$t_{we}$	Write Enable To Output Delay			40	ns	
$t_{set}$	Data Setup Time	15			ns	
$t_h$	Data Hold Time	20			ns	
$t_r$	Reset To Output Delay			40	ns	
$t_s$	Set To Output Delay			30	ns	
$t_e$	Output Enable/Disable Time			45	ns	
$t_c$	Clear To Output Delay			55	ns	

**CAPACITANCE\***
 $F = 1\text{ MHz}, V_{BIAS} = 2.5\text{V}, V_{CC} = +5\text{V}, T_A = 25^\circ\text{C}$ 

Symbol	Test	LIMITS	
		Typ.	Max.
$C_{IN}$	$DS_1$ MD Input Capacitance	9 pF	12 pF
$C_{IN}$	$DS_2, CK, ACK, DI_1-DI_8$ Input Capacitance	5 pF	9 pF
$C_{OUT}$	$DO_1-DO_8$ Output Capacitance	8 pF	12 pF

\*This parameter is sampled and not 100% tested.

**SWITCHING CHARACTERISTICS****Conditions of Test**

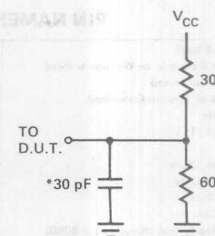
Input Pulse Amplitude = 2.5 V

Input Rise and Fall Times 5 ns

Between 1V and 2V Measurements made at 1.5V with 15 mA & 30 pF Test Load

**Test Load**

15mA & 30pF



\* INCLUDING JIG & PROBE CAPACITANCE

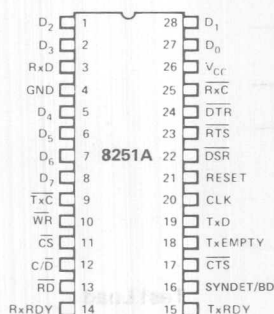


## 8251A PROGRAMMABLE COMMUNICATION INTERFACE

- Synchronous and Asynchronous Operation
- Synchronous 5-8 Bit Characters; Internal or External Character Synchronization; Automatic Sync Insertion
- Asynchronous 5-8 Bit Characters; Clock Rate—1, 16 or 64 Times Baud Rate; Break Character Generation; 1, 1½, or 2 Stop Bits; False Start Bit Detection; Automatic Break Detect and Handling; 19.2K Baud.
- Baud Rate — DC to 64K Baud
- Full Duplex, Double Buffered, Transmitter and Receiver
- Error Detection — Parity, Overrun and Framing
- Fully Compatible with 8080/8085 CPU
- 28-Pin DIP Package
- All Inputs and Outputs are TTL Compatible
- Single +5V Supply
- Single TTL Clock

The Intel® 8251A is the enhanced version of the industry standard, Intel® 8251 Universal Synchronous/Asynchronous Receiver/Transmitter (USART), designed for data communications with Intel's new high performance family of microprocessors such as the 8085. The 8251A is used as a peripheral device and is programmed by the CPU to operate using virtually any serial data transmission technique presently in use (including IBM "bi-sync"). The USART accepts data characters from the CPU in parallel format and then converts them into a continuous serial data stream for transmission. Simultaneously, it can receive serial data streams and convert them into parallel data characters for the CPU. The USART will signal the CPU whenever it can accept a new character for transmission or whenever it has received a character for the CPU. The CPU can read the complete status of the USART at any time. These include data transmission errors and control signals such as SYNDET, TxEMPTY. The chip is constructed using N-channel silicon gate technology.

PIN CONFIGURATION

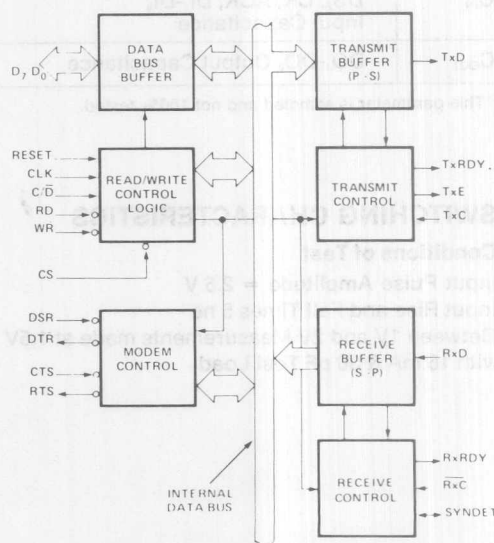


PIN NAMES

D <sub>7</sub> /D <sub>0</sub>	Data Bus (8 bits)
C/D	Control or Data is to be Written or Read
RD	Read Data Command
WR	Write Data or Control Command
CS	Chip Enable
CLK	Clock Pulse (TTL)
RESET	Reset
TxC	Transmitter Clock
TxD	Transmitter Data
RxC	Receiver Clock
RxD	Receiver Data
RxRDY	Receiver Ready (has character for 8080)
TxRDY	Transmitter Ready (ready for char. from 8080)

DSR	Data Set Ready
DTR	Data Terminal Ready
SYNDET/BD	Sync Detect/ Break Detect
RTS	Request to Send Data
CTS	Clear to Send Data
TxE	Transmitter Empty
V <sub>CC</sub>	+5 Volt Supply
GND	Ground

BLOCK DIAGRAM



## FEATURES AND ENHANCEMENTS

8251A is an advanced design of the industry standard USART, the Intel® 8251. The 8251A operates with an extended range of Intel microprocessors that includes the new 8085 CPU and maintains compatibility with the 8251. Familiarization time is minimal because of compatibility and involves only knowing the additional features and enhancements, and reviewing the AC and DC specifications of the 8251A.

The 8251A incorporates all the key features of the 8251 and has the following additional features and enhancements:

- 8251A has double-buffered data paths with separate I/O registers for control, status, Data In, and Data Out, which considerably simplifies control programming and minimizes CPU overhead.
- In asynchronous operations, the Receiver detects and handles "break" automatically, relieving the CPU of this task.
- A refined Rx initialization prevents the Receiver from starting when in "break" state, preventing unwanted interrupts from a disconnected USART.
- At the conclusion of a transmission, TxD line will always return to the marking state unless SBRK is programmed.

- Tx Enable logic enhancement prevents a Tx Disable command from halting transmission until all data previously written has been transmitted. The logic also prevents the transmitter from turning off in the middle of a word.
- When External Sync Detect is programmed, Internal Sync Detect is disabled, and an External Sync Detect status is provided via a flip-flop which clears itself upon a status read.
- Possibility of false sync detect is minimized by ensuring that if double character sync is programmed, the characters be contiguously detected and also by clearing the Rx register to all ones whenever Enter Hunt command is issued in Sync mode.
- As long as the 8251A is not selected, the RD and WR do not affect the internal operation of the device.
- The 8251A Status can be read at any time but the status update will be inhibited during status read.
- The 8251A is free from extraneous glitches and has enhanced AC and DC characteristics, providing higher speed and better operating margins.
- Baud rate from DC to 64K.
- Fully compatible with Intel's new industry standard, the MCS-85.

	CS	RD	WR	IO/M
DATA IN - DATA OUT	0	1	0	0
DATA IN - DATA OUT	0	1	0	0
DATA IN - DATA OUT	0	1	0	0
DATA IN - DATA OUT	0	1	0	0
DATA IN - DATA OUT	0	1	0	0
DATA IN - DATA OUT	0	1	0	0
DATA IN - DATA OUT	0	1	0	0
DATA IN - DATA OUT	0	1	0	0

## 8251A BASIC FUNCTIONAL DESCRIPTION

### General

The 8251A is a Universal Synchronous/Asynchronous Receiver/Transmitter designed specifically for the 80/85 Microcomputer Systems. Like other I/O devices in a Microcomputer System, its functional configuration is programmed by the system's software for maximum flexibility. The 8251A can support virtually any serial data technique currently in use including bi-sync.

In a communication environment an interface device must convert parallel format system data into serial format for transmission and convert incoming serial format data into parallel system data for reception. The interface device must also delete or insert bits or characters that are functionally unique to the communication technique. In essence, the interface should appear "transparent" to the CPU, a simple input or output of byte-oriented system data.

### Data Bus Buffer

This 3-state, bidirectional, 8-bit buffer is used to interface the 8251A to the system Data Bus. Data is transmitted or received by the buffer upon execution of INPUT or OUTPUT instructions of the CPU. Control words, Command words and Status information are also transferred through the Data Bus Buffer. The command status and data in, and data out are separate 8-bit registers to provide double buffering.

This functional block accepts inputs from the system Control bus and generates control signals for overall device operation. It contains the Control Word Register and Command Word Register that store the various control formats for the device functional definition.

### RESET (Reset)

A "high" on this input forces the 8251A into an "Idle" mode. The device will remain at "Idle" until a new set of control words is written into the 8251A to program its functional definition. Minimum RESET pulse width is  $6 t_{CY}$  (clock must be running).

### CLK (Clock)

The CLK input is used to generate internal device timing and is normally connected to the Phase 2 (TTL) output of the 8224 Clock Generator. No external inputs or outputs are referenced to CLK but the frequency of CLK must be greater than 30 times the Receiver or Transmitter data bit rates.

### WR (Write)

A "low" on this input informs the 8251A that the CPU is writing data or control words to the 8251A.

### RD (Read)

A "low" on this input informs the 8251A that the CPU is reading data or status information from the 8251A.

### C/D (Control/Data)

This input, in conjunction with the  $\overline{WR}$  and  $\overline{RD}$  inputs, informs the 8251A that the word on the Data Bus is either a data character, control word or status information.

1 = CONTROL/STATUS 0 = DATA

### CS (Chip Select)

A "low" on this input selects the 8251A. No reading or writing will occur unless the device is selected. When  $\overline{CS}$  is high, the Data Bus in the float state and  $\overline{RD}$  and  $\overline{WR}$  will have no effect on the chip.

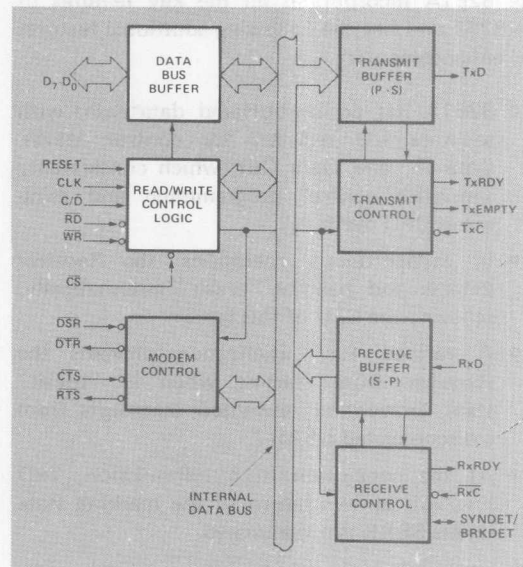


Figure 1. 8251A Block Diagram Showing Data Bus Buffer and Read/Write Logic Functions

C/D	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	
0	0	1	0	8251A DATA $\Rightarrow$ DATA BUS
0	1	0	0	DATA BUS $\Rightarrow$ 8251A DATA
1	0	1	0	STATUS $\Rightarrow$ DATA BUS
1	1	0	0	DATA BUS $\Rightarrow$ CONTROL
X	1	1	0	DATA BUS $\Rightarrow$ 3-STATE
X	X	X	1	DATA BUS $\Rightarrow$ 3-STATE

### Modem Control

The 8251A has a set of control inputs and outputs that can be used to simplify the interface to almost any modem. The modem control signals are general purpose in nature and can be used for functions other than modem control, if necessary.

**DSR (Data Set Ready)**

The  $\overline{\text{DSR}}$  input signal is a general purpose, 1-bit inverting input port. Its condition can be tested by the CPU using a Status Read operation. The  $\overline{\text{DSR}}$  input is normally used to test modem conditions such as Data Set Ready.

**DTR (Data Terminal Ready)**

The  $\overline{\text{DTR}}$  output signal is a general purpose, 1-bit inverting output port. It can be set "low" by programming the appropriate bit in the Command Instruction word. The  $\overline{\text{DTR}}$  output signal is normally used for modem control such as Data Terminal Ready or Rate Select.

**RTS (Request to Send)**

The  $\overline{\text{RTS}}$  output signal is a general purpose, 1-bit inverting output port. It can be set "low" by programming the appropriate bit in the Command Instruction word. The  $\overline{\text{RTS}}$  output signal is normally used for Modem control such as Request to Send.

**CTS (Clear to Send)**

A "low" on this input enables the 8251A to transmit serial data if the Tx Enable bit in the Command byte is set to a "one." If either a Tx Enable off or CTS off condition occurs while the Tx is in operation, the Tx will transmit all the data in the USART, written prior to Tx Disable command before shutting down.

**Transmitter Buffer**

The Transmitter Buffer accepts parallel data from the Data Bus Buffer, converts it to a serial bit stream, inserts the appropriate characters or bits (based on the communication technique) and outputs a composite serial stream of data on the  $\text{TxD}$  output pin on the falling edge of  $\overline{\text{TxC}}$ . The transmitter will begin transmission upon being enabled if  $\text{CTS} = 0$ . The  $\text{TxD}$  line will be held in the marking state immediately upon a master Reset or when Tx Enable/CTS off or TxEMPTY.

**Transmitter Control**

The transmitter Control manages all activities associated with the transmission of serial data. It accepts and issues signals both externally and internally to accomplish this function.

**TxRDY (Transmitter Ready)**

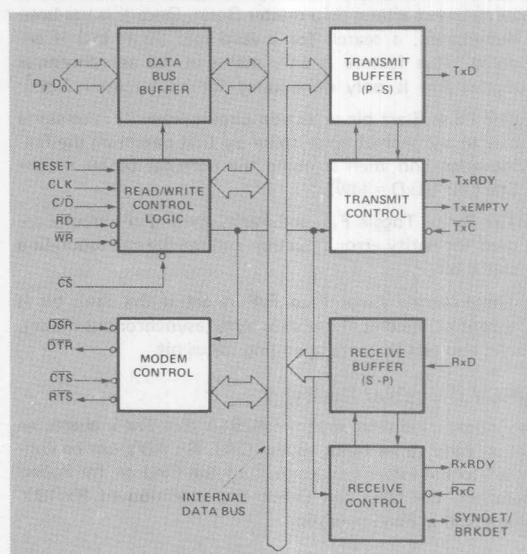
This output signals the CPU that the transmitter is ready to accept a data character. The  $\text{TxRDY}$  output pin can be used as an interrupt to the system, since it is masked by Tx Disabled, or, for Polled operation, the CPU can check  $\text{TxRDY}$  using a Status Read operation.  $\text{TxRDY}$  is automatically reset by the leading edge of  $\overline{\text{WR}}$  when a data character is loaded from the CPU.

Note that when using the Polled operation, the  $\text{TxRDY}$  status bit is *not* masked by Tx Enabled, but will only indicate the Empty/Full Status of the Tx Data Input Register.

**TxE (Transmitter Empty)**

When the 8251A has no characters to transmit, the  $\text{TxEMPTY}$  output will go "high". It resets automatically upon receiving a character from the CPU.  $\text{TxEMPTY}$  can be used to indicate the end of a transmission mode, so that the CPU "knows" when to "turn the line around" in the half-duplexed operational mode.  $\text{TxEMPTY}$  is independent of the Tx Enable bit in the Command instruction.

In SYNChronous mode, a "high" on this output indicates that a character has not been loaded and the SYNC character or characters are about to be or are being transmitted automatically as "fillers".  $\text{TxEMPTY}$  does not go low when the SYNC characters are being shifted out.



**Figure 2. 8251A Block Diagram Showing Modem and Transmitter Buffer and Control Functions**

**TxC (Transmitter Clock)**

The Transmitter Clock controls the rate at which the character is to be transmitted. In the Synchronous transmission mode, the Baud Rate (1x) is equal to the  $\text{TxC}$  frequency. In Asynchronous transmission mode the baud rate is a fraction of the actual  $\text{TxC}$  frequency. A portion of the mode instruction selects this factor; it can be 1, 1/16 or 1/64 the  $\text{TxC}$ .

For Example:

If Baud Rate equals 110 Baud,  
 $\overline{\text{TxC}}$  equals 110 Hz (1x)  
 $\overline{\text{TxC}}$  equals 1.76 kHz (16x)  
 $\overline{\text{TxC}}$  equals 7.04 kHz (64x).

The falling edge of  $\overline{\text{TxC}}$  shifts the serial data out of the 8251A.

to parallel format, checks for bits or characters that are unique to the communication technique and sends an "assembled" character to the CPU. Serial data is input to RxD pin, and is clocked in on the rising edge of  $\overline{\text{RxC}}$ .

### Receiver Control

This functional block manages all receiver-related activities which consist of the following features:

The RxD initialization circuit prevents the 8251A from mistaking an unused input line for an active low data line in the "break condition". Before starting to receive serial characters on the RxD line, a valid "1" must first be detected after a chip master Reset. Once this has been determined, a search for a valid low (Start bit) is enabled. This feature is only active in the asynchronous mode, and is only done once for each master Reset.

The False Start bit detection circuit prevents false starts due to a transient noise spike by first detecting the falling edge and then strobing the nominal center of the Start bit (RxD = low).

The Parity Toggle F/F and Parity Error F/F circuits are used for parity error detection and set the corresponding status bit.

The Framing Error Flag F/F is set if the Stop bit is absent at the end of the data byte (asynchronous mode), and also sets the corresponding status bit.

### RxRDY (Receiver Ready)

This output indicates that the 8251A contains a character that is ready to be input to the CPU. Rx RDY can be connected to the interrupt structure of the CPU or, for Polled operation, the CPU can check the condition of RxRDY using a Status Read operation.

Rx Enable off both masks and holds RxRDY in the Reset Condition. For Asynchronous mode, to set RxRDY, the Receiver must be Enabled to sense a Start Bit and a complete character must be assembled and transferred to the Data Output Register. For Synchronous mode, to set RxRDY, the Receiver must be enabled and a character must finish assembly and be transferred to the Data Output Register.

Failure to read the received character from the Rx Data Output Register prior to the assembly of the next Rx Data character will set overrun condition error and the previous character will be written over and lost. If the Rx Data is being read by the CPU when the internal transfer is occurring, overrun error will be set and the old character will be lost.

### RxC (Receiver Clock)

The Receiver Clock controls the rate at which the character is to be received. In Synchronous Mode, the Baud Rate (1x) is equal to the actual frequency of  $\overline{\text{RxC}}$ . In Asynchronous Mode, the Baud Rate is a fraction of the actual  $\overline{\text{RxC}}$  fre-

For Example:

Baud Rate equals 300 Baud, if

$\overline{\text{RxC}}$  equals 300 Hz (1x)

$\overline{\text{RxC}}$  equals 4800 Hz (16x)

$\overline{\text{RxC}}$  equals 19.2 kHz (64x).

Baud Rate equals 2400 Baud, if

$\overline{\text{RxC}}$  equals 2400 Hz (1x)

$\overline{\text{RxC}}$  equals 38.4 kHz (16x)

$\overline{\text{RxC}}$  equals 153.6 kHz (64x).

Data is sampled into the 8251A on the rising edge of  $\overline{\text{RxC}}$ .

**NOTE:** In most communications systems, the 8251A will be handling both the transmission and reception operations of a single link. Consequently, the Receive and Transmit Baud Rates will be the same. Both  $\overline{\text{TxC}}$  and  $\overline{\text{RxC}}$  will require identical frequencies for this operation and can be tied together and connected to a single frequency source (Baud Rate Generator) to simplify the interface.

### SYNDET (SYNC Detect)/BRKDET (Break Detect)

This pin is used in SYNChronous Mode for SYNDET and may be used as either input or output, programmable through the Control Word. It is reset to output mode low upon RESET. When used as an output (internal Sync mode), the SYNDET pin will go "high" to indicate that the 8251A has located the SYNC character in the Receive mode. If the 8251A is programmed to use double Sync characters (bi-sync), then SYNDET will go "high" in the middle of the last bit of the second Sync character. SYNDET is automatically reset upon a Status Read operation.

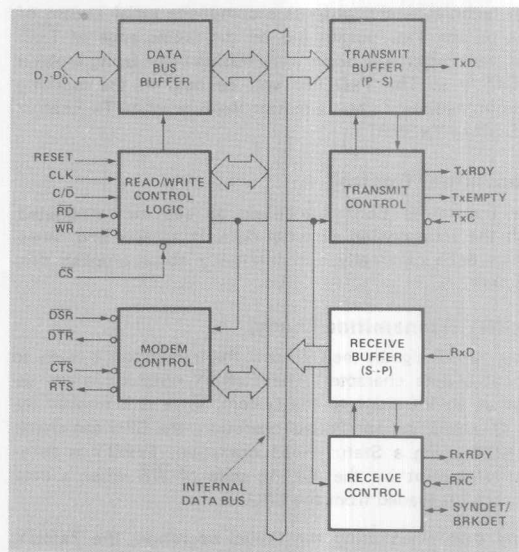


Figure 3. 8251A Block Diagram Showing Receiver Buffer and Control Functions

When used as an input (external SYNC detect mode), a positive going signal will cause the 8251A to start assembling data characters on the rising edge of the next  $\overline{\text{RxC}}$ . Once in SYNC, the "high" input signal can be removed. the period of  $\overline{\text{RxC}}$ . When External SYNC Detect is programmed, the Internal SYNC Detect is disabled.

## BREAK DETECT (Async Mode Only)

This output will go high whenever an all zero word of the programmed length (including start bit, data bit, parity bit, and one stop bit) is received. Break Detect may also be read as a Status bit. It is reset only upon a master chip Reset or Rx Data returning to a "one" state.

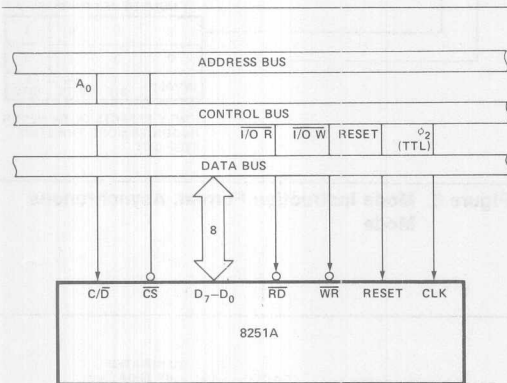


Figure 4. 8251A Interface to 8080 Standard System Bus

## DETAILED OPERATION DESCRIPTION

### General

The complete functional definition of the 8251A is programmed by the system's software. A set of control words must be sent out by the CPU to initialize the 8251A to support the desired communications format. These control words will program the: BAUD RATE, CHARACTER LENGTH, NUMBER OF STOP BITS, SYNCHRONOUS or ASYNCHRONOUS OPERATION, EVEN/ODD/OFF PARITY, etc. In the Synchronous Mode, options are also provided to select either internal or external character synchronization.

Once programmed, the 8251A is ready to perform its communication functions. The TxRDY output is raised "high" to signal the CPU that the 8251A is ready to receive a data character from the CPU. This output (TxRDY) is reset automatically when the CPU writes a character into the 8251A. On the other hand, the 8251A receives serial data from the MODEM or I/O device. Upon receiving an entire character, the RxRDY output is raised "high" to signal the CPU that the 8251A has a complete character ready for the CPU to fetch. RxRDY is reset automatically upon the CPU data read operation.

The 8251A cannot begin transmission until the Tx Enable (Transmitter Enable) bit is set in the Command Instruction and it has received a Clear To Send (CTS) input. The Tx/D output will be held in the marking state upon Reset.

### Programming the 8251A

Prior to starting data transmission or reception, the 8251A must be loaded with a set of control words generated by the CPU. These control signals define the complete functional definition of the 8251A and must immediately follow a Reset operation (internal or external).

The control words are split into two formats:

1. Mode Instruction
2. Command Instruction

### Mode Instruction

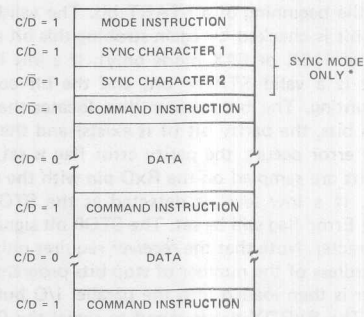
This format defines the general operational characteristics of the 8251A. It must follow a Reset operation (internal or external). Once the Mode Instruction has been written into the 8251A by the CPU, SYNC characters or Command Instructions may be inserted.

### Command Instruction

This format defines a status word that is used to control the actual operation of the 8251A.

Both the Mode and Command Instructions must conform to a specified sequence for proper device operation. The Mode Instruction must be inserted immediately following a Reset operation, prior to using the 8251A for data communication.

All control words written into the 8251A after the Mode Instruction will load the Command Instruction. Command Instructions can be written into the 8251A at any time in the data block during the operation of the 8251A. To return to the Mode Instruction format, the master Reset bit in the Command Instruction word can be set to initiate an internal Reset operation which automatically places the 8251A back into the Mode Instruction format. Command Instructions must follow the Mode Instructions or Sync characters.



\* The second SYNC character is skipped if MODE instruction has programmed the 8251A to single character Internal SYNC Mode. Both SYNC characters are skipped if MODE instruction has programmed the 8251A to ASYNC mode.

Figure 5. Typical Data Block

### Mode Instruction Definition

The 8251A can be used for either Asynchronous or Synchronous data communication. To understand how the Mode Instruction defines the functional operation of the 8251A, the designer can best view the device as two separate components sharing the same package, one Asynchronous the other Synchronous. The format definition can be changed only after a master chip Reset. For explanation purposes the two formats will be isolated.

**NOTE:** When parity is enabled it is not considered as one of the data bits for the purpose of programming the word length. The actual parity bit received on the Rx Data line cannot be read on the Data Bus. In the case of a programmed character length of less than 8 bits, the least significant Data Bus bits will hold the data; unused bits are "don't care" when writing data to the 8251A, and will be "zeros" when reading the data from the 8251A.

### Asynchronous Mode (Transmission)

Whenever a data character is sent by the CPU the 8251A automatically adds a Start bit (low level) followed by the data bits (least significant bit first), and the programmed number of Stop bits to each character. Also, an even or odd Parity bit is inserted prior to the Stop bit(s), as defined by the Mode Instruction. The character is then transmitted as a serial data stream on the TxD output. The serial data is shifted out on the falling edge of  $\overline{\text{TxC}}$  at a rate equal to 1, 1/16, or 1/64 that of the  $\overline{\text{TxC}}$ , as defined by the Mode Instruction. BREAK characters can be continuously sent to the TxD if commanded to do so.

When no data characters have been loaded into the 8251A the TxD output remains "high" (marking) unless a Break (continuously low) has been programmed.

### Asynchronous Mode (Receive)

The RxD line is normally high. A falling edge on this line triggers the beginning of a START bit. The validity of this START bit is checked by again strobing this bit at its nominal center (16X or 64X mode only). If a low is detected again, it is a valid START bit, and the bit counter will start counting. The bit counter thus locates the center of the data bits, the parity bit (if it exists) and the stop bits. If parity error occurs, the parity error flag is set. Data and parity bits are sampled on the RxD pin with the rising edge of  $\overline{\text{RxC}}$ . If a low level is detected as the STOP bit, the Framing Error flag will be set. The STOP bit signals the end of a character. Note that the receiver requires only one stop bit, regardless of the number of stop bits programmed. This character is then loaded into the parallel I/O buffer of the 8251A. The RxRDY pin is raised to signal the CPU that a character is ready to be fetched. If a previous character has not been fetched by the CPU, the present character replaces it in the I/O buffer, and the OVERRUN Error flag is raised (thus the previous character is lost). All of the error flags can be reset by an Error Reset Instruction. The occurrence of any of these errors will not affect the operation of the 8251A.

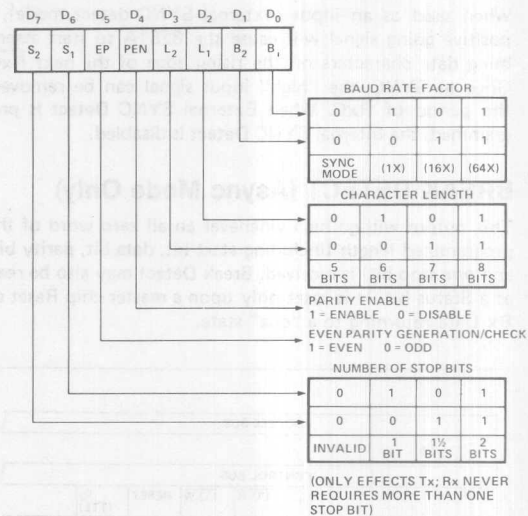


Figure 6. Mode Instruction Format, Asynchronous Mode

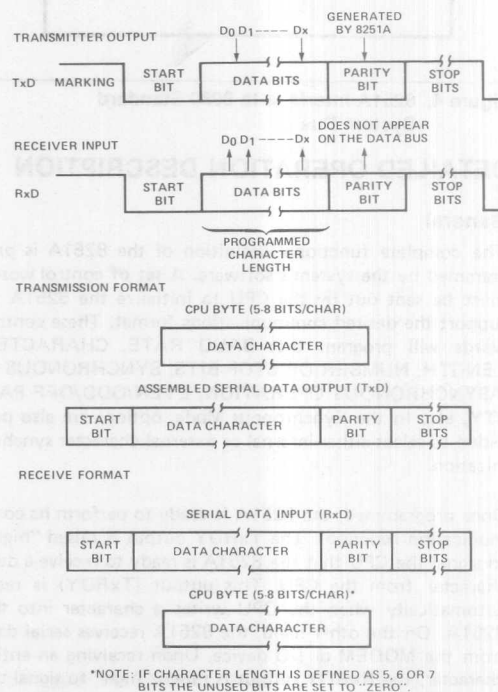
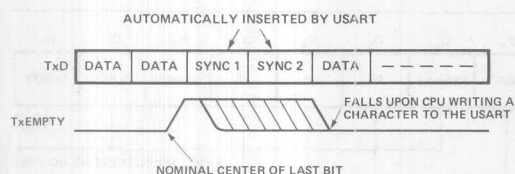


Figure 7. Asynchronous Mode

### Synchronous Mode (Transmission)

The TxD output is continuously high until the CPU sends its first character to the 8251A which usually is a SYNC character. When the  $\overline{\text{CTS}}$  line goes low, the first character is serially transmitted out. All characters are shifted out on the falling edge of  $\overline{\text{TxC}}$ . Data is shifted out at the same rate as the  $\overline{\text{TxC}}$ .

Once transmission has started, the data stream at the TxD output must continue at the  $\overline{\text{TxC}}$  rate. If the CPU does not provide the 8251A with a data character before the 8251A Transmitter Buffers become empty, the SYNC characters (or character if in single SYNC character mode) will be automatically inserted in the TxD data stream. In this case, the TxEMPTY pin is raised high to signal that the 8251A is empty and SYNC characters are being sent out. TxEMPTY does not go low when the SYNC is being shifted out (see figure below). The TxEMPTY pin is internally reset by a data character being written into the 8251A.



### Synchronous Mode (Receive)

In this mode, character synchronization can be internally or externally achieved. If the SYNC mode has been programmed, ENTER HUNT command should be included in the first command instruction word written. Data on the Rx pin is then sampled in on the rising edge of  $\overline{\text{RxC}}$ . The content of the Rx buffer is compared at every bit boundary with the first SYNC character until a match occurs. If the 8251A has been programmed for two SYNC characters, the subsequent received character is also compared; when both SYNC characters have been detected, the USART ends the HUNT mode and is in character synchronization. The SYNDT pin is then set high, and is reset automatically by a STATUS READ. If parity is programmed, SYNDT will not be set until the middle of the parity bit instead of the middle of the last data bit.

In the external SYNC mode, synchronization is achieved by applying a high level on the SYNDT pin, thus forcing the 8251A out of the HUNT mode. The high level can be removed after one  $\overline{\text{RxC}}$  cycle. An ENTER HUNT command has no effect in the asynchronous mode of operation.

Parity error and overrun error are both checked in the same way as in the Asynchronous Rx mode. Parity is checked when not in Hunt, regardless of whether the Receiver is enabled or not.

The CPU can command the receiver to enter the HUNT mode if synchronization is lost. This will also set all the used character bits in the buffer to a "one", thus preventing a possible false SYNDT caused by data that happens to be in the Rx Buffer at ENTER HUNT time. Note that

the SYNDT F/F is reset at each Status Read, regardless of whether internal or external SYNC has been programmed. This does not cause the 8251A to return to the HUNT mode. When in SYNC mode, but not in HUNT, Sync Detection is still functional, but only occurs at the "known" word boundaries. Thus, if one Status Read indicates SYNDT and a second Status Read also indicates SYNDT, then the programmed SYNDT characters have been received since the previous Status Read. (If double character sync has been programmed, then both sync characters have been contiguously received to gate a SYNDT indication.) When external SYNDT mode is selected, internal Sync Detect is disabled, and the SYNDT F/F may be set at any bit boundary.

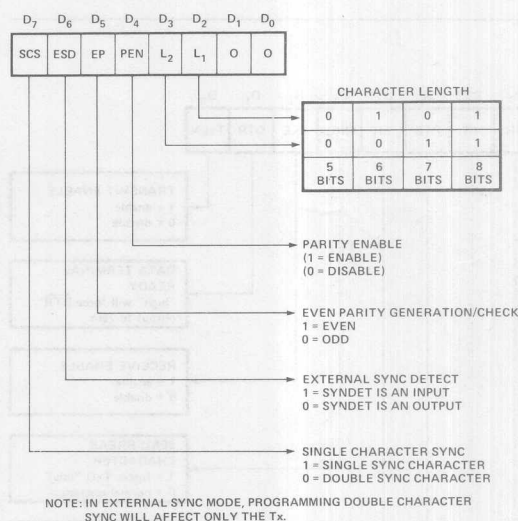


Figure 8. Mode Instruction Format

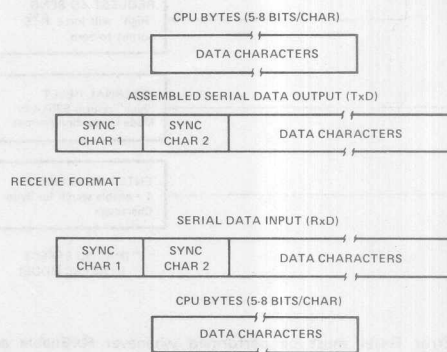
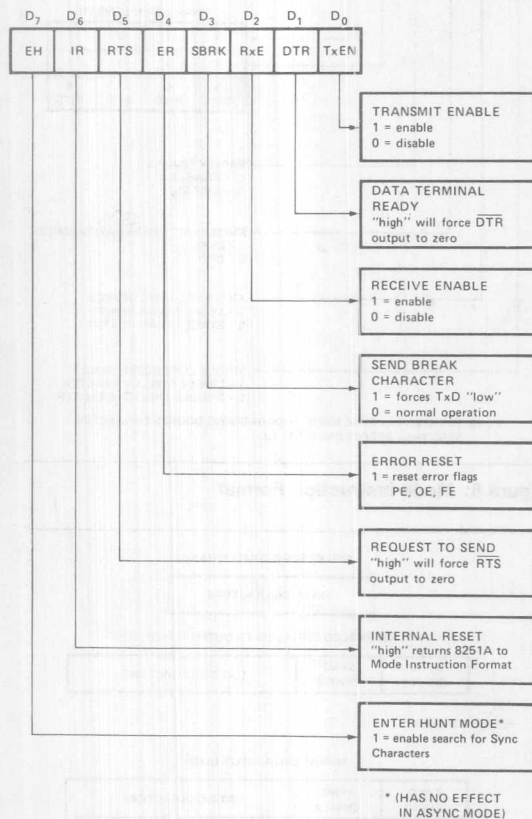


Figure 9. Data Format, Synchronous Mode

specified by the Mode Instruction and the Sync Characters are loaded (if in Sync Mode) then the device is ready to be used for data communication. The Command Instruction controls the actual operation of the selected format. Functions such as: Enable Transmit/Receive, Error Reset and Modem Controls are provided by the Command Instruction.

Once the Mode Instruction has been written into the 8251A and Sync characters inserted, if necessary, then all further "control writes" ( $C/\bar{D} = 1$ ) will load a Command Instruction. A Reset Operation (internal or external) will return the 8251A to the Mode Instruction format.



Note: Error Reset must be performed whenever RxEnable and Enter Hunt are programmed.

Figure 10. Command Instruction Format

## 8251A

examine the "status" of the active device to ascertain if errors have occurred or other conditions that require the processor's attention. The 8251A has facilities that allow the programmer to "read" the status of the device at any time during the functional operation. (The status update is inhibited during status read).

A normal "read" command is issued by the CPU with  $C/\bar{D} = 1$  to accomplish this function.

Some of the bits in the Status Read Format have identical meanings to external output pins so that the 8251A can be used in a completely Polled environment or in an interrupt driven environment. TxRDY is an exception.

Note that status update can have a maximum delay of 28 clock periods from the actual event affecting the status.

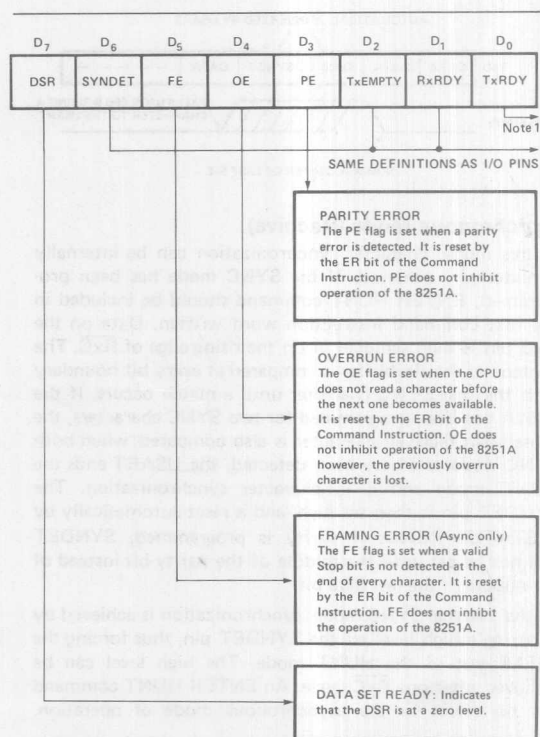


Figure 11. Status Read Format

## APPLICATIONS OF THE 8251A

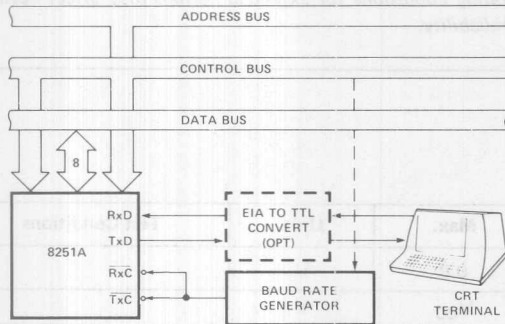


Figure 12. Asynchronous Serial Interface to CRT Terminal, DC-9600 Baud

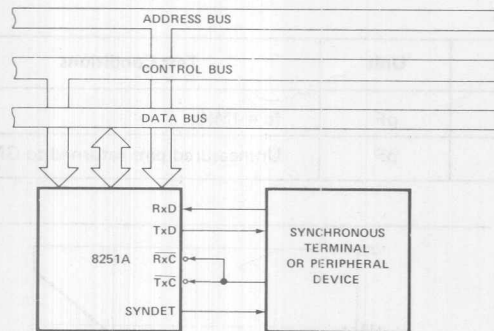


Figure 13. Synchronous Interface to Terminal or Peripheral Device

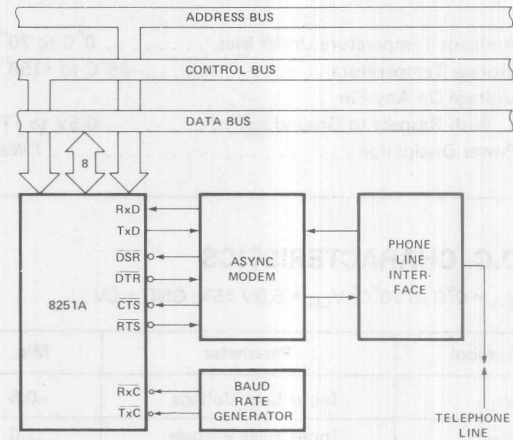


Figure 14. Asynchronous Interface to Telephone Lines

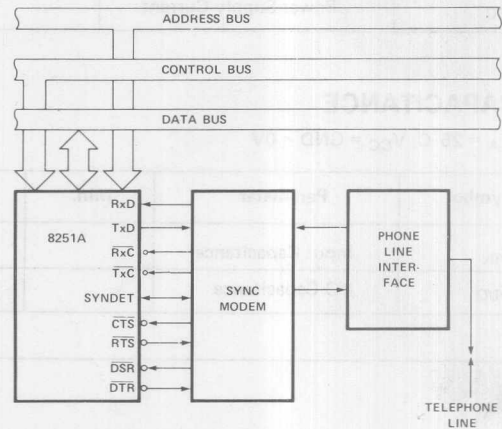


Figure 15. Synchronous Interface to Telephone Lines

**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature Under Bias. . . . . 0°C to 70°C  
 Storage Temperature . . . . . -65°C to +150°C  
 Voltage On Any Pin  
     With Respect to Ground. . . . . -0.5V to +7V  
 Power Dissipation . . . . . 1 Watt

\*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

**D.C. CHARACTERISTICS**

$T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ;  $V_{CC} = 5.0\text{V} \pm 5\%$ ;  $\text{GND} = 0\text{V}$

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$V_{IL}$	Input Low Voltage	-0.5	0.8	V	
$V_{IH}$	Input High Voltage	2.0	$V_{CC}$	V	
$V_{OL}$	Output Low Voltage		0.45	V	$I_{OL} = 2.2\text{ mA}$
$V_{OH}$	Output High Voltage	2.4		V	$I_{OH} = -400\text{ }\mu\text{A}$
$I_{OFL}$	Output Float Leakage		$\pm 10$	$\mu\text{A}$	$V_{OUT} = V_{CC}$ TO $0.45\text{V}$
$I_{IL}$	Input Leakage		$\pm 10$	$\mu\text{A}$	$V_{IN} = V_{CC}$ TO $0.45\text{V}$
$I_{CC}$	Power Supply Current		100	mA	All Outputs = High

**CAPACITANCE**

$T_A = 25^\circ\text{C}$ ;  $V_{CC} = \text{GND} = 0\text{V}$

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$C_{IN}$	Input Capacitance		10	pF	$f_c = 1\text{MHz}$
$C_{I/O}$	I/O Capacitance		20	pF	Unmeasured pins returned to GND

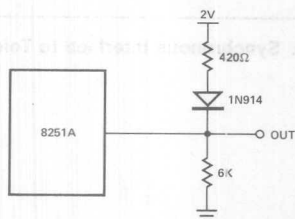


Figure 16. Test Load Circuit

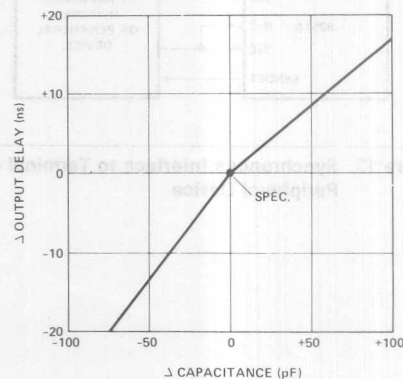


Figure 17. Typical  $\Delta$  Output Delay vs.  $\Delta$  Capacitance (pF)

## A.C. CHARACTERISTICS

$T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ;  $V_{CC} = 5.0\text{V} \pm 5\%$ ;  $\text{GND} = 0\text{V}$

## Bus Parameters (Note 1)

## Read Cycle:

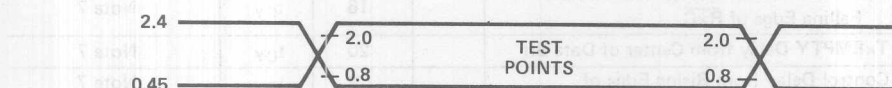
SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
$t_{AR}$	Address Stable Before $\overline{\text{READ}}$ ( $\overline{\text{CS}}$ , $\text{C}/\overline{\text{D}}$ )	0		ns	Note 2
$t_{RA}$	Address Hold Time for $\overline{\text{READ}}$ ( $\overline{\text{CS}}$ , $\text{C}/\overline{\text{D}}$ )	0		ns	Note 2
$t_{RR}$	$\overline{\text{READ}}$ Pulse Width	250		ns	
$t_{RD}$	Data Delay from $\overline{\text{READ}}$		200	ns	3, $C_L = 150\text{ pF}$
$t_{DF}$	$\overline{\text{READ}}$ to Data Floating	10	100	ns	

## Write Cycle:

SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
$t_{AW}$	Address Stable Before $\overline{\text{WRITE}}$	0		ns	
$t_{WA}$	Address Hold Time for $\overline{\text{WRITE}}$	0		ns	
$t_{WW}$	$\overline{\text{WRITE}}$ Pulse Width	250		ns	
$t_{DW}$	Data Set Up Time for $\overline{\text{WRITE}}$	150		ns	
$t_{WD}$	Data Hold Time for $\overline{\text{WRITE}}$	0		ns	
$t_{RV}$	Recovery Time Between WRITES	6		$t_{CY}$	Note 4

- NOTES: 1. AC timings measured  $V_{OH} = 2.0$ ,  $V_{OL} = 0.8$ , and with load circuit of Figure 1.  
 2. Chip Select ( $\overline{\text{CS}}$ ) and Command/Data ( $\text{C}/\overline{\text{D}}$ ) are considered as Addresses.  
 3. Assumes that Address is valid before  $\overline{\text{RD}}\downarrow$ .  
 4. This recovery time is for Mode Initialization only. Write Data is allowed only when  $\text{TxRDY} = 1$ .  
 Recovery Time between Writes for Asynchronous Mode is  $8 t_{CY}$  and for Synchronous Mode is  $16 t_{CY}$ .

## Input Waveforms for AC Tests



# Other Timings:

SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
$t_{CY}$	Clock Period	320	1.35	$\mu s$	Notes 5, 6
$t_{\phi}$	Clock High Pulse Width	120	$t_{CY-90}$	ns	
$t_{\bar{\phi}}$	Clock Low Pulse Width	90		ns	
$t_R, t_F$	Clock Rise and Fall Time	5	20	ns	
$t_{DTx}$	TxD Delay from Falling Edge of $\bar{TxC}$		1	$\mu s$	
$t_{SRx}$	Rx Data Set-Up Time to Sampling Pulse	2		$\mu s$	
$t_{HRx}$	Rx Data Hold Time to Sampling Pulse	2		$\mu s$	
$f_{Tx}$	Transmitter Input Clock Frequency				
	1x Baud Rate	DC	64	kHz	
	16x Baud Rate	DC	310	kHz	
	64x Baud Rate	DC	615	kHz	
$t_{TPW}$	Transmitter Input Clock Pulse Width				
	1x Baud Rate	12		$t_{CY}$	
	16x and 64x Baud Rate	1		$t_{CY}$	
$t_{TPD}$	Transmitter Input Clock Pulse Delay				
	1x Baud Rate	15		$t_{CY}$	
	16x and 64x Baud Rate	3		$t_{CY}$	
$f_{Rx}$	Receiver Input Clock Frequency				
	1x Baud Rate	DC	64	kHz	
	16x Baud Rate	DC	310	kHz	
	64x Baud Rate	DC	615	kHz	
$t_{RPW}$	Receiver Input Clock Pulse Width				
	1x Baud Rate	12		$t_{CY}$	
	16x and 64x Baud Rate	1		$t_{CY}$	
$t_{RPD}$	Receiver Input Clock Pulse Delay				
	1x Baud Rate	15		$t_{CY}$	
	16x and 64x Baud Rate	3		$t_{CY}$	
$t_{TxRDY}$	TxRDY Pin Delay from Center of last Bit		8	$t_{CY}$	Note 7
$t_{TxRDY\ CLEAR}$	TxRDY $\downarrow$ from Leading Edge of $\bar{WR}$		150	ns	Note 7
$t_{RxRDY}$	RxRDY Pin Delay from Center of last Bit		24	$t_{CY}$	Note 7
$t_{RxRDY\ CLEAR}$	RxRDY $\downarrow$ from Leading Edge of $\bar{RD}$		150	ns	Note 7
$t_{IS}$	Internal SYND $\bar{E}$ T Delay from Rising Edge of $\bar{RxC}$		24	$t_{CY}$	Note 7
$t_{ES}$	External SYND $\bar{E}$ T Set-Up Time Before Falling Edge of $\bar{RxC}$		16	$t_{CY}$	Note 7
$t_{TxEMPTY}$	TxEMPTY Delay from Center of Data Bit		20	$t_{CY}$	Note 7
$t_{WC}$	Control Delay from Rising Edge of WRITE (TxEn, $\bar{DTR}$ , RTS)		8	$t_{CY}$	Note 7
$t_{CR}$	Control to READ Set-Up Time ( $\bar{DSR}$ , $\bar{CTS}$ )		20	$t_{CY}$	Note 7

5. The Tx $\bar{C}$  and Rx $\bar{C}$  frequencies have the following limitations with respect to CLK.

For 1x Baud Rate,  $f_{Tx}$  or  $f_{Rx} \leq 1/(30 t_{CY})$

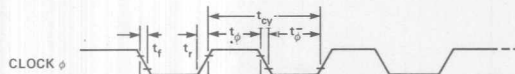
For 16x and 64x Baud Rate,  $f_{Tx}$  or  $f_{Rx} \leq 1/(4.5 t_{CY})$

6. Reset Pulse Width = 6  $t_{CY}$  minimum; System Clock must be running during Reset.

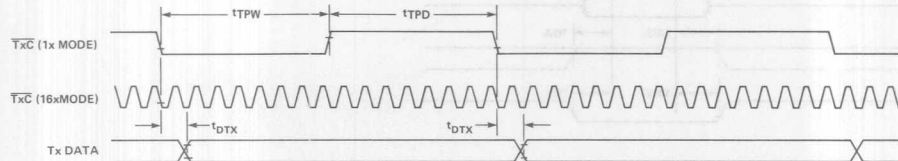
7. Status update can have a maximum delay of 28 clock periods from the event affecting the status.

## WAVEFORMS

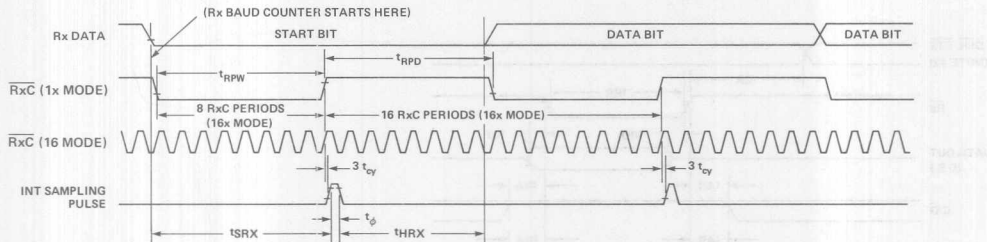
### System Clock Input



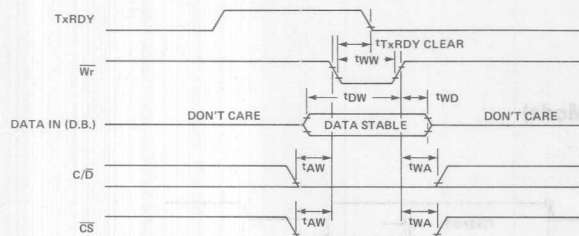
### Transmitter Clock & Data



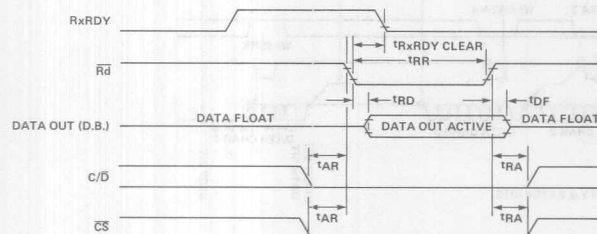
### Receiver Clock & Data



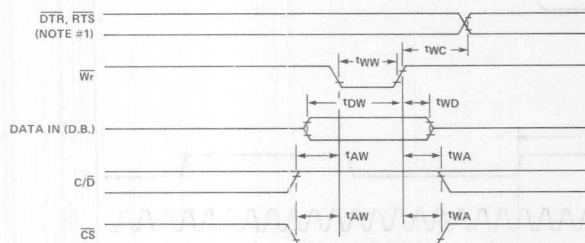
### Write Data Cycle (CPU → USART)



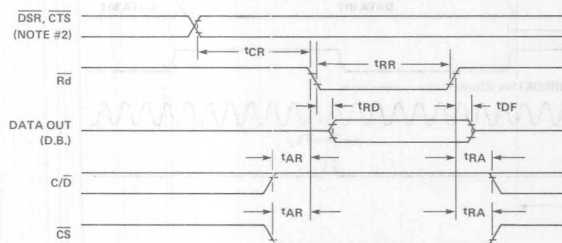
### Read Data Cycle (CPU ← USART)



## Write Control or Output Port Cycle (CPU → USART)

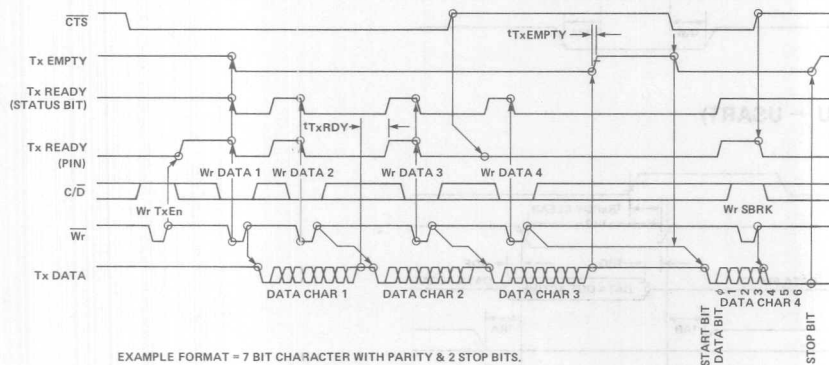


## Read Control or Input Port (CPU ← USART)



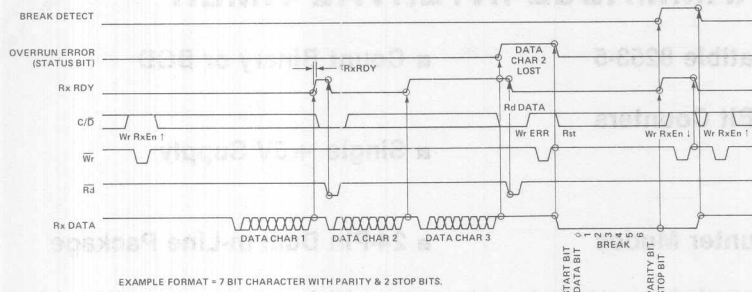
NOTE #1:  $t_{WC}$  INCLUDES THE RESPONSE TIMING OF A CONTROL BYTE.  
 NOTE #2:  $t_{CR}$  INCLUDES THE EFFECT OF CTS ON THE  $\overline{TxENBL}$  CIRCUITRY.

## Transmitter Control &amp; Flag Timing (ASYNC Mode)

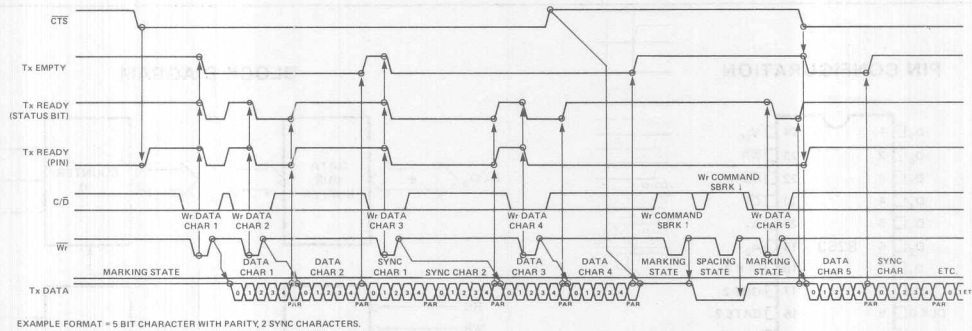


EXAMPLE FORMAT = 7 BIT CHARACTER WITH PARITY & 2 STOP BITS.

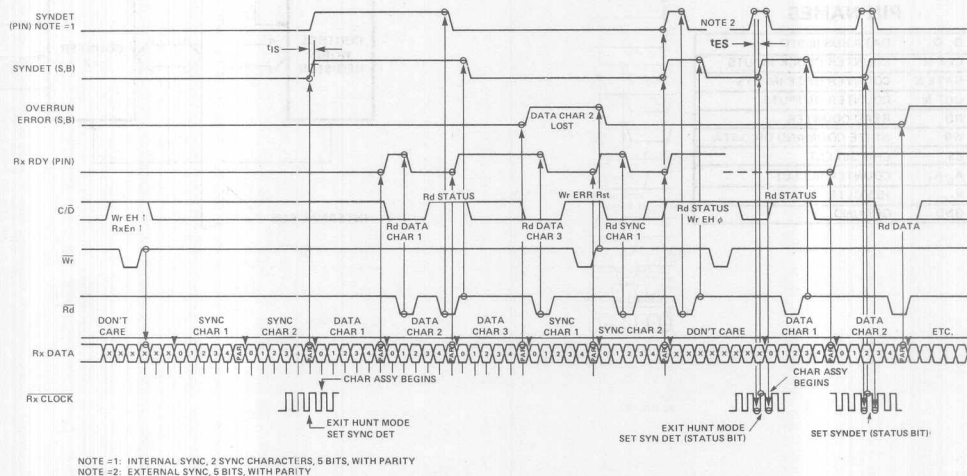
## Receiver Control &amp; Flag Timing (ASync Mode)



## Transmitter Control &amp; Flag Timing (SYNC Mode)



## Receiver Control &amp; Flag Timing (SYNC Mode)



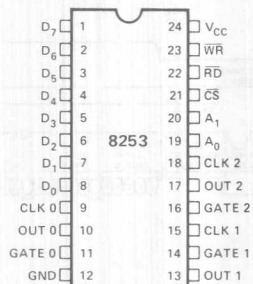
## 8253/8253-5 PROGRAMMABLE INTERVAL TIMER

- MCS—85™ Compatible 8253-5
- Count Binary or BCD
- 3 Independent 16-Bit Counters
- Single +5V Supply
- DC to 2 MHz
- Programmable Counter Modes
- 24-Pin Dual In-Line Package

The Intel® 8253 is a programmable counter/timer chip designed for use as an Intel microcomputer peripheral. It uses nMOS technology with a single +5V supply and is packaged in a 24-pin plastic DIP.

It is organized as 3 independent 16-bit counters, each with a count rate of up to 2 MHz. All modes of operation are software programmable.

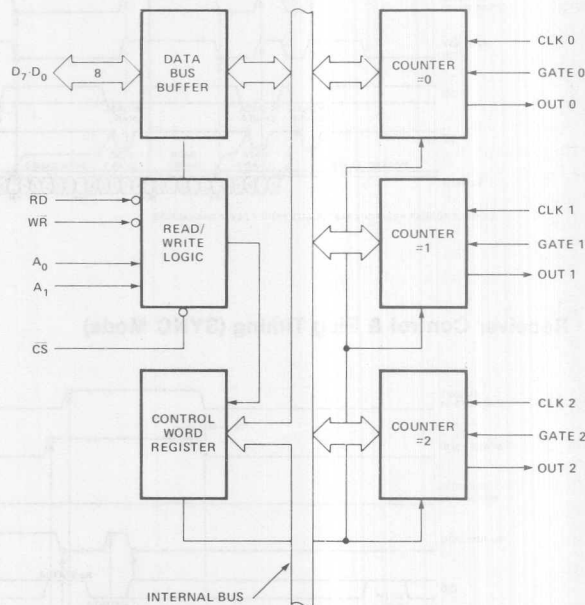
### PIN CONFIGURATION



### PIN NAMES

D <sub>7</sub> -D <sub>0</sub>	DATA BUS (8 BIT)
CLK N	COUNTER CLOCK INPUTS
GATE N	COUNTER GATE INPUTS
OUT N	COUNTER OUTPUTS
RD	READ COUNTER
WR	WRITE COMMAND OR DATA
CS	CHIP SELECT
A <sub>0</sub> -A <sub>1</sub>	COUNTER SELECT
V <sub>CC</sub>	+5 VOLTS
GND	GROUND

### BLOCK DIAGRAM



## FUNCTIONAL DESCRIPTION

### General

The 8253 is a programmable interval timer/counter specifically designed for use with the Intel™ Microcomputer systems. Its function is that of a general purpose, multi-timing element that can be treated as an array of I/O ports in the system software.

The 8253 solves one of the most common problems in any microcomputer system, the generation of accurate time delays under software control. Instead of setting up timing loops in systems software, the programmer configures the 8253 to match his requirements, initializes one of the counters of the 8253 with the desired quantity, then upon command the 8253 will count out the delay and interrupt the CPU when it has completed its tasks. It is easy to see that the software overhead is minimal and that multiple delays can easily be maintained by assignment of priority levels.

Other counter/timer functions that are non-delay in nature but also common to most microcomputers can be implemented with the 8253.

- Programmable Rate Generator
- Event Counter
- Binary Rate Multiplier
- Real Time Clock
- Digital One-Shot
- Complex Motor Controller

### Data Bus Buffer

This 3-state, bi-directional, 8-bit buffer is used to interface the 8253 to the system data bus. Data is transmitted or received by the buffer upon execution of INput or OUTput CPU instructions. The Data Bus Buffer has three basic functions.

1. Programming the MODES of the 8253.
2. Loading the count registers.
3. Reading the count values.

### Read/Write Logic

The Read/Write Logic accepts inputs from the system bus and in turn generates control signals for overall device operation. It is enabled or disabled by CS so that no operation can occur to change the function unless the device has been selected by the system logic.

### RD (Read)

A "low" on this input informs the 8253 that the CPU is inputting data in the form of a counters value.

### WR (Write)

A "low" on this input informs the 8253 that the CPU is outputting data in the form of mode information or loading counters.

### A0, A1

These inputs are normally connected to the address bus. Their function is to select one of the three counters to be operated on and to address the control word register for mode selection.

### CS (Chip Select)

A "low" on this input enables the 8253. No reading or writing will occur unless the device is selected. The CS input has no effect upon the actual operation of the counters.

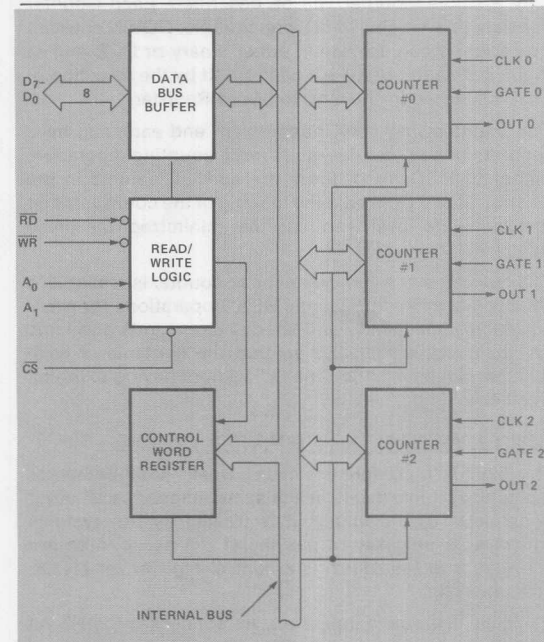


Figure 1. Block Diagram Showing Data Bus Buffer and Read/Write Logic Functions

CS	RD	WR	A <sub>1</sub>	A <sub>0</sub>	
0	1	0	0	0	Load Counter No. 0
0	1	0	0	1	Load Counter No. 1
0	1	0	1	0	Load Counter No. 2
0	1	0	1	1	Write Mode Word
0	0	1	0	0	Read Counter No. 0
0	0	1	0	1	Read Counter No. 1
0	0	1	1	0	Read Counter No. 2
0	0	1	1	1	No-Operation 3-State
1	X	X	X	X	Disable 3-State
0	1	1	X	X	No-Operation 3-State

### Control Word Register

The Control Word Register is selected when A<sub>0</sub>, A<sub>1</sub> are 11. It then accepts information from the data bus buffer and stores it in a register. The information stored in this register controls the operational MODE of each counter, selection of binary or BCD counting and the loading of each count register.

The Control Word Register can only be written into; no read operation of its contents is available.

### Counter #0, Counter #1, Counter #2

These three functional blocks are identical in operation so only a single Counter will be described. Each Counter consists of a single, 16-bit, pre-settable, DOWN counter. The counter can operate in either binary or BCD and its input, gate and output are configured by the selection of MODES stored in the Control Word Register.

The counters are fully independent and each can have separate Mode configuration and counting operation, binary or BCD. Also, there are special features in the control word that handle the loading of the count value so that software overhead can be minimized for these functions.

The reading of the contents of each counter is available to the programmer with simple READ operations for event counting applications and special commands and logic are included in the 8253 so that the contents of each counter can be read "on the fly" without having to inhibit the clock input.

### 8253 SYSTEM INTERFACE

The 8253 is a component of the Intel™ Microcomputer Systems and interfaces in the same manner as all other peripherals of the family. It is treated by the systems software as an array of peripheral I/O ports; three are counters and the fourth is a control register for MODE programming.

Basically, the select inputs A<sub>0</sub>, A<sub>1</sub> connect to the A<sub>0</sub>, A<sub>1</sub> address bus signals of the CPU. The CS can be derived directly from the address bus using a linear select method. Or, it can be connected to the output of a decoder, such as an Intel® 8205 for larger systems.

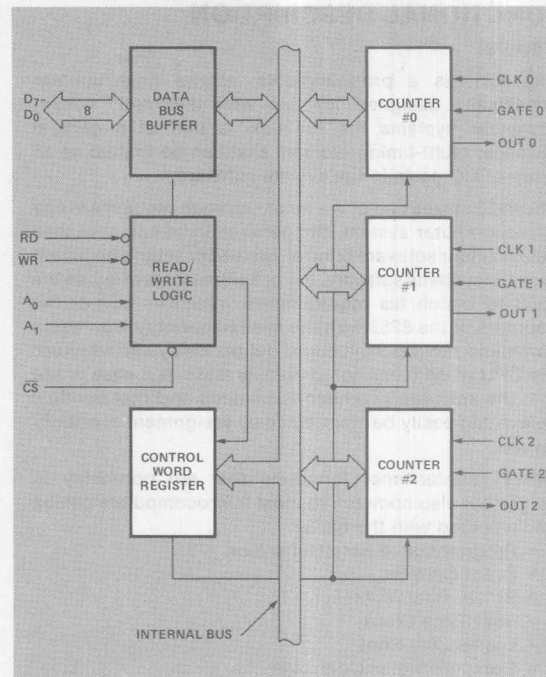


Figure 2. Block Diagram Showing Control Word Register and Counter Functions

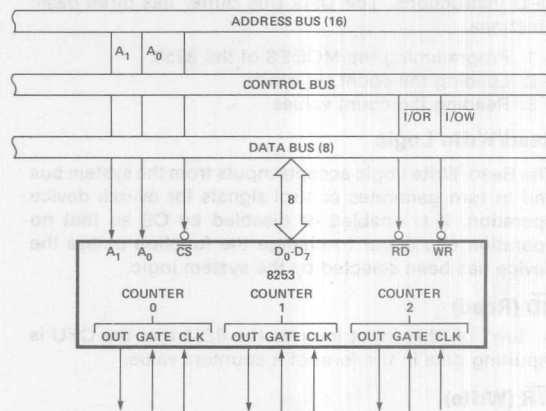


Figure 3. 8253 System Interface

## OPERATIONAL DESCRIPTION

### General

The complete functional definition of the 8253 is programmed by the systems software. A set of control words must be sent out by the CPU to initialize each counter of the 8253 with the desired MODE and quantity information. These control words program the MODE, Loading sequence and selection of binary or BCD counting.

Once programmed, the 8253 is ready to perform whatever timing tasks it is assigned to accomplish.

The actual counting operation of each counter is completely independent and additional logic is provided on-chip so that the usual problems associated with efficient monitoring and management of external, asynchronous events or rates to the microcomputer system have been eliminated.

### Programming the 8253

All of the MODES for each counter are programmed by the systems software by simple I/O operations.

Each counter of the 8253 is individually programmed by writing a control word into the Control Word Register. (A0, A1 = 11)

### Control Word Format

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
SC1	SC0	RL1	RL0	M2	M1	M0	BCD

### Definition of Control

#### SC — Select Counter:

SC1	SC0	
0	0	Select Counter 0
0	1	Select Counter 1
1	0	Select Counter 2
1	1	Illegal

#### RL — Read/Load:

RL1	RL0	
0	0	Counter Latching operation (see READ/WRITE Procedure Section)
1	0	Read/Load most significant byte only.
0	1	Read/Load least significant byte only.
1	1	Read/Load least significant byte first, then most significant byte.

#### M — MODE:

M2	M1	M0	
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

#### BCD:

0	Binary Counter 16-bits
1	Binary Coded Decimal (BCD) Counter (4 Decades)

### MODE Definition

**MODE: Interrupt on Terminal Count.** The output will be initially low after the mode set operation. After the count is loaded into the selected count register, the output will remain low and the counter will count. When terminal count is reached the output will go high and remain high until the selected count register is reloaded with the mode or a new count is loaded.

Reloading a counter register during counting results in the following:

- (1) Load 1st byte stops the current counting.
- (2) Load 2nd byte starts the new count.

**MODE 1: Programmable One-Shot.** The output will go low on the count following the rising edge of the gate input.

The output will go high on the terminal count. If a new count value is loaded while the output is low it will not affect the duration of the one-shot pulse until the succeeding trigger. The current count can be read at any time without affecting the one-shot pulse.

The one-shot is retriggerable, hence the output will remain low for the full count after any rising edge of the gate input.

**MODE 2: Rate Generator.** Divide by N counter. The output will be low for one period of the input clock. The period from one output pulse to the next equals the number of input counts in the count register. If the count register is reloaded between output pulses the present period will not be affected, but the subsequent period will reflect the new value.

The gate input, when low, will force the output high. When the gate input goes high, the counter will start from the initial count. Thus, the gate input can be used to synchronize the counter.

When this mode is set, the output will remain high until after the count register is loaded. The output then can also be synchronized by software.

**MODE 3: Square Wave Rate Generator.** Similar to MODE 2 except that the output will remain high until one half the count has been completed (for even numbers) and go low for the other half of the count. If the count is odd, the output will be high for  $(N + 1)/2$  counts and low for  $(N - 1)/2$  counts.

If the counter register is reloaded with a new value during counting, this new value will be reflected immediately after the output transition of the current count.

**MODE 4: Software Triggered Strobe.** After the mode is set, the output will be high. When the count is loaded, the counter will begin counting. On terminal count, the output will go low for one input clock period, then will go high again.

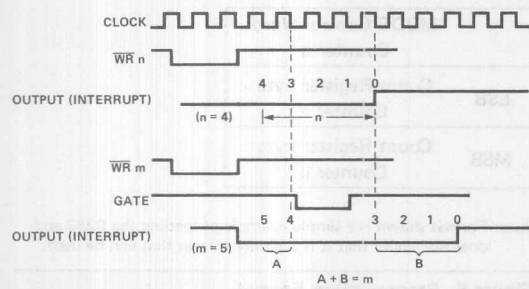
If the count register is reloaded between output pulses the present period will not be affected, but the subsequent period will reflect the new value. The count will be inhibited while the gate input is low. Reloading the counter register will restart counting beginning with the new number.

**MODE 5: Hardware Triggered Strobe.** The counter will start counting after the rising edge of the trigger input and will go low for one clock period when the terminal count is reached. The counter is retriggerable. The output will not go low until the full count after the rising edge of any trigger.

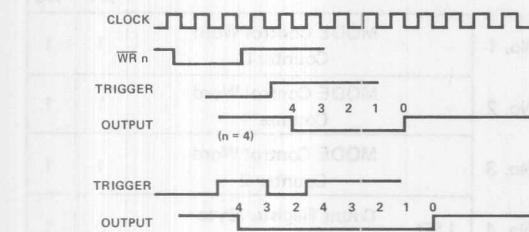
Signal Status	Low Or Going Low	Rising	High
0	Disables counting	---	Enables counting
1	---	1) Initiates counting 2) Resets output after next clock	---
2	1) Disables counting 2) Sets output immediately high	Initiates counting	Enables counting
3	1) Disables counting 2) Sets output immediately high	Initiates counting	Enables counting
4	Disables counting	---	Enables counting
5	---	Initiates counting	---

Figure 4. Gate Pin Operations Summary

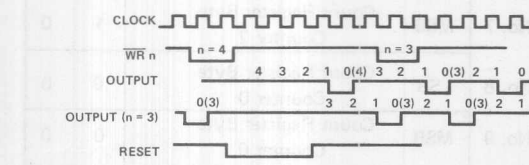
## MODE 0



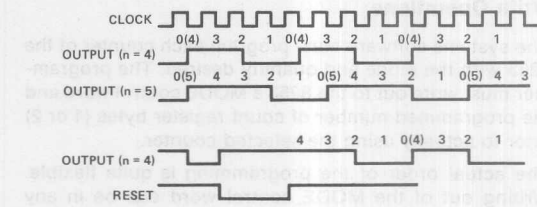
## MODE 1



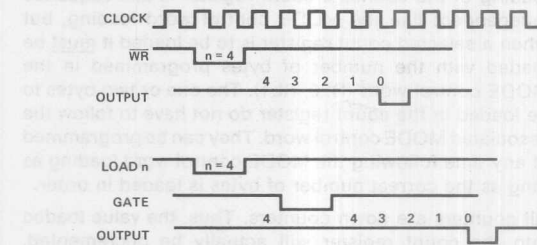
## MODE 2



## MODE 3



## MODE 4



## MODE 5

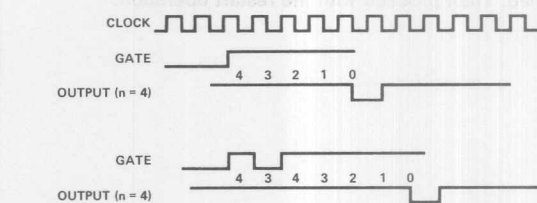


Figure 5. 8253 Timing Diagrams

## 8253 READ/WRITE PROCEDURE

## Write Operations

The systems software must program each counter of the 8253 with the mode and quantity desired. The programmer must write out to the 8253 a MODE control word and the programmed number of count register bytes (1 or 2) prior to actually using the selected counter.

The actual order of the programming is quite flexible. Writing out of the MODE control word can be in any sequence of counter selection, e.g., counter #0 does not have to be first or counter #2 last. Each counter's MODE control word register has a separate address so that its loading is completely sequence independent. (SC0, SC1)

The loading of the Count Register with the actual count value, however, must be done in exactly the sequence programmed in the MODE control word (RL0, RL1). This loading of the counter's count register is still sequence independent like the MODE control word loading, but when a selected count register is to be loaded it must be loaded with the number of bytes programmed in the MODE control word (RL0, RL1). The one or two bytes to be loaded in the count register do not have to follow the associated MODE control word. They can be programmed at any time following the MODE control word loading as long as the correct number of bytes is loaded in order.

All counters are down counters. Thus, the value loaded into the count register will actually be decremented. Loading all zeroes into a count register will result in the maximum count ( $2^{16}$  for Binary or  $10^4$  for BCD). In MODE 0 the new count will not restart until the load has been completed. It will accept one of two bytes depending on how the MODE control words (RL0, RL1) are programmed. Then proceed with the restart operation.

MODE Control Word	
Counter n	
LSB	Count Register byte Counter n
MSB	Count Register byte Counter n

Note: Format shown is a simple example of loading the 8253 and does not imply that it is the only format that can be used.

Figure 6. Programming Format

		A1	A0
No. 1	MODE Control Word Counter 0	1	1
No. 2	MODE Control Word Counter 1	1	1
No. 3	MODE Control Word Counter 2	1	1
No. 4	LSB Count Register Byte Counter 1	0	1
No. 5	MSB Count Register Byte Counter 1	0	1
No. 6	LSB Count Register Byte Counter 2	1	0
No. 7	MSB Count Register Byte Counter 2	1	0
No. 8	LSB Count Register Byte Counter 0	0	0
No. 9	MSB Count Register Byte Counter 0	0	0

Note: The exclusive addresses of each counter's count register make the task of programming the 8253 a very simple matter, and maximum effective use of the device will result if this feature is fully utilized.

Figure 7. Alternate Programming Formats

## Read Operations

In most counter applications it becomes necessary to read the value of the count in progress and make a computational decision based on this quantity. Event counters are probably the most common application that uses this function. The 8253 contains logic that will allow the programmer to easily read the contents of any of the three counters without disturbing the actual count in progress.

There are two methods that the programmer can use to read the value of the counters. The first method involves the use of simple I/O read operations of the selected counter. By controlling the A0, A1 inputs to the 8253 the programmer can select the counter to be read (remember that no read operation of the mode register is allowed A0, A1-11). The only requirement with this method is that in order to assure a stable count reading the actual operation of the selected counter must be inhibited either by controlling the Gate input or by external logic that inhibits the clock input. The contents of the counter selected will be available as follows:

- first I/O Read contains the least significant byte (LSB).
- second I/O Read contains the most significant byte (MSB).

Due to the internal logic of the 8253 it is absolutely necessary to complete the entire reading procedure. If two bytes are programmed to be read then two bytes must be read before any loading WR command can be sent to the same counter.

## Read Operation Chart

A1	A0	RD	
0	0	0	Read Counter No. 0
0	1	0	Read Counter No. 1
1	0	0	Read Counter No. 2
1	1	0	Illegal

## Reading While Counting

In order for the programmer to read the contents of any counter without effecting or disturbing the counting operation the 8253 has special internal logic that can be accessed using simple WR commands to the MODE register. Basically, when the programmer wishes to read the contents of a selected counter "on the fly" he loads the MODE register with a special code which latches the present count value into a storage register so that its contents contain an accurate, stable quantity. The programmer then issues a normal read command to the selected counter and the contents of the latched register is available.

## MODE Register for Latching Count

A0, A1 = 11

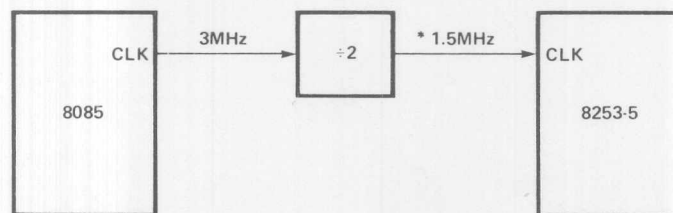
D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	0	0	X	X	X	X

SC1, SC0 — specify counter to be latched.

D5, D4 — 00 designates counter latching operation.

X — don't care.

The same limitation applies to this mode of reading the counter as the previous method. That is, it is mandatory to complete the entire read operation as programmed.



\*If an 8085 clock output is to drive an 8253-5 clock input, it must be reduced to 2 MHz or less.

Figure 8. MCS-85™ Clock Interface\*

**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature Under Bias	0°C to 70°C
Storage Temperature	-65°C to +150°C
Voltage On Any Pin	
With Respect to Ground	-0.5 V to +7 V
Power Dissipation	1 Watt

\*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

**D.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ;  $V_{CC} = 5\text{V} \pm 5\%$ )

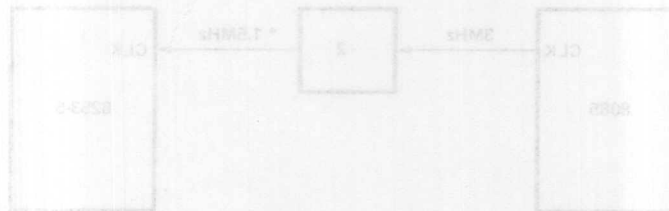
SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
$V_{IL}$	Input Low Voltage	-0.5	0.8	V	
$V_{IH}$	Input High Voltage	2.2	$V_{CC} + 0.5\text{V}$	V	
$V_{OL}$	Output Low Voltage		0.45	V	Note 1
$V_{OH}$	Output High Voltage	2.4		V	Note 2
$I_{IL}$	Input Load Current		$\pm 10$	$\mu\text{A}$	$V_{IN} = V_{CC}$ to 0V
$I_{OFL}$	Output Float Leakage		$\pm 10$	$\mu\text{A}$	$V_{OUT} = V_{CC}$ to 0V
$I_{CC}$	$V_{CC}$ Supply Current		140	mA	

Note 1: 8253,  $I_{OL} = 1.6\text{ mA}$ ; 8253-5,  $I_{OL} = 2.2\text{ mA}$ .

Note 2: 8253,  $I_{OH} = -150\text{ }\mu\text{A}$ ; 8253-5,  $I_{OH} = -400\text{ }\mu\text{A}$ .

**CAPACITANCE**  $T_A = 25^\circ\text{C}$ ;  $V_{CC} = \text{GND} = 0\text{V}$ 

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
$C_{IN}$	Input Capacitance			10	pF	$f_c = 1\text{ MHz}$
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured pins returned to $V_{SS}$



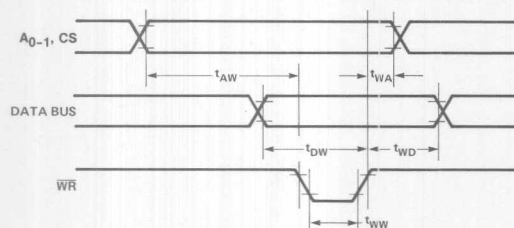
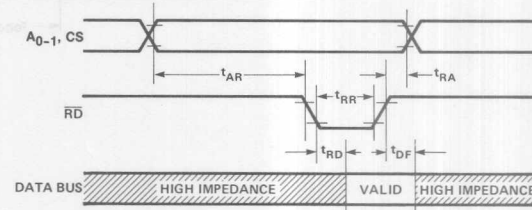
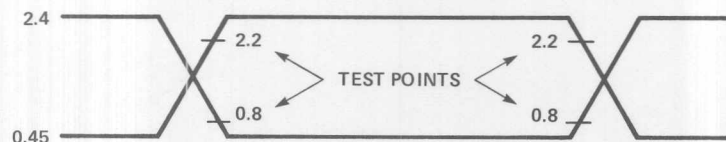
**A.C. CHARACTERISTICS**  $T_A = 0^\circ\text{C to } 70^\circ\text{C}$ ;  $V_{CC} = 5.0\text{V} \pm 5\%$ ;  $\text{GND} = 0\text{V}$ **Bus Parameters (Note 1)****Read Cycle:**

SYMBOL	PARAMETER	8253		8253-5		UNIT
		MIN.	MAX.	MIN.	MAX.	
$t_{AR}$	Address Stable Before $\overline{\text{READ}}$	50		50		ns
$t_{RA}$	Address Hold Time for $\overline{\text{READ}}$	5		5		ns
$t_{RR}$	$\overline{\text{READ}}$ Pulse Width	400		300		ns
$t_{RD}$	Data Delay From $\overline{\text{READ}}$ <sup>[2]</sup>		300		200	ns
$t_{DF}$	$\overline{\text{READ}}$ to Data Floating	25	125	25	100	ns

**Write Cycle:**

SYMBOL	PARAMETER	8253		8253-5		UNIT
		MIN.	MAX.	MIN.	MAX.	
$t_{AW}$	Address Stable Before $\overline{\text{WRITE}}$	50		50		ns
$t_{WA}$	Address Hold Time for $\overline{\text{WRITE}}$	30		30		ns
$t_{WW}$	$\overline{\text{WRITE}}$ Pulse Width	400		300		ns
$t_{DW}$	Data Set Up Time for $\overline{\text{WRITE}}$	300		250		ns
$t_{WD}$	Data Hold Time for $\overline{\text{WRITE}}$	40		30		ns
$t_{RV}$	Recovery Time Between $\overline{\text{WRITES}}$	1		1		$\mu\text{s}$

Notes: 1. AC timings measured at  $V_{OH} = 2.2$ ,  $V_{OL} = 0.8$   
 2. Test Conditions: 8253,  $C_L = 100\text{pF}$ ; 8253-5:  $C_L = 150\text{pF}$ .

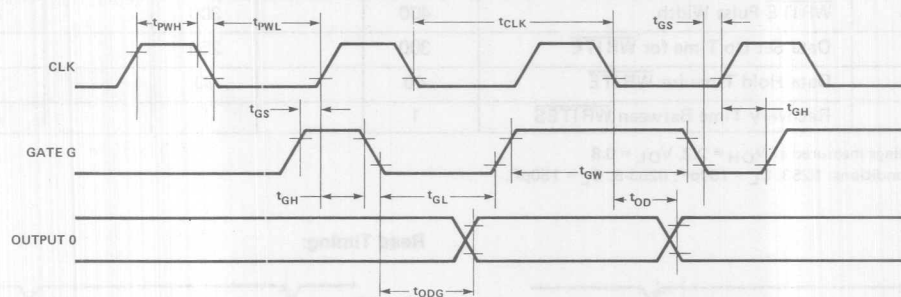
**Write Timing:****Read Timing:****Input Waveforms for A.C. Tests:**

**PRELIMINARY**  
 Notice: This is not a final specification. Some  
 parametric limits are subject to change.

## Clock and Gate Timing:

SYMBOL	PARAMETER	8253		8253-5		UNIT
		MIN.	MAX.	MIN.	MAX.	
$t_{CLK}$	Clock Period	380	dc	380	dc	ns
$t_{PWH}$	High Pulse Width	230		230		ns
$t_{PWL}$	Low Pulse Width	150		150		ns
$t_{GW}$	Gate Width High	150		150		ns
$t_{GL}$	Gate Width Low	100		100		ns
$t_{GS}$	Gate Set Up Time to CLK $\uparrow$	100		100		ns
$t_{GH}$	Gate Hold Time After CLK $\uparrow$	50		50		ns
$t_{OD}$	Output Delay From CLK $\downarrow$ <sup>[1]</sup>		400		400	ns
$t_{ODG}$	Output Delay From Gate $\downarrow$ <sup>[1]</sup>		300		300	ns

Note 1: Test Conditions: 8253:  $C_L = 100\text{pF}$ ; 8253-5:  $C_L = 150\text{pF}$ .



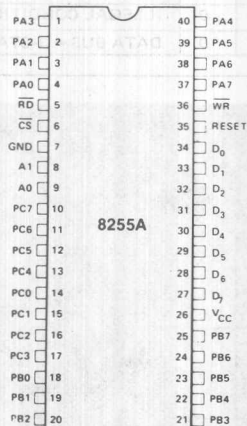


## 8255A/8255A-5 PROGRAMMABLE PERIPHERAL INTERFACE

- MCS-85™ Compatible 8255A-5
- 24 Programmable I/O Pins
- Completely TTL Compatible
- Fully Compatible with Intel® Microprocessor Families
- Improved Timing Characteristics
- Direct Bit Set/Reset Capability Easing Control Application Interface
- 40-Pin Dual In-Line Package
- Reduces System Package Count
- Improved DC Driving Capability

The Intel® 8255A is a general purpose programmable I/O device designed for use with Intel® microprocessors. It has 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation. In the first mode (MODE 0), each group of 12 I/O pins may be programmed in sets of 4 to be input or output. In MODE 1, the second mode, each group may be programmed to have 8 lines of input or output. Of the remaining 4 pins, 3 are used for handshaking and interrupt control signals. The third mode of operation (MODE 2) is a bidirectional bus mode which uses 8 lines for a bidirectional bus, and 5 lines, borrowing one from the other group, for handshaking.

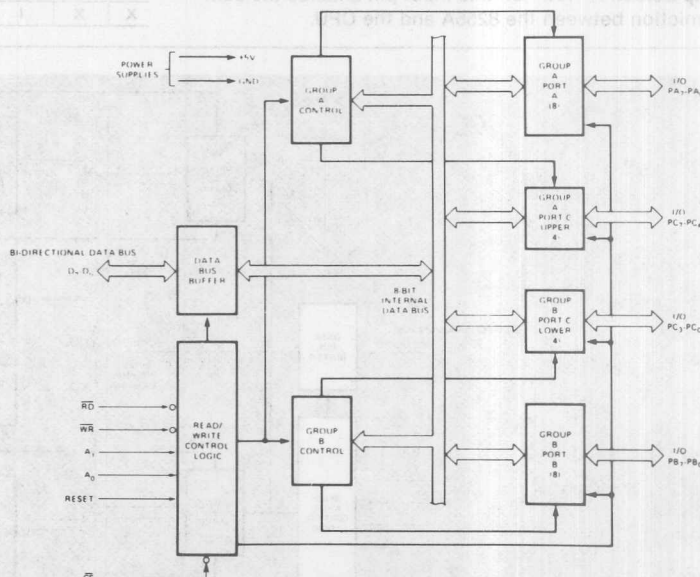
### PIN CONFIGURATION



### PIN NAMES

D <sub>7</sub> -D <sub>0</sub>	DATA BUS (BI-DIRECTIONAL)
RESET	RESET INPUT
CS	CHIP SELECT
RD	READ INPUT
WR	WRITE INPUT
A <sub>0</sub> , A <sub>1</sub>	PORT ADDRESS
PA <sub>7</sub> -PA <sub>0</sub>	PORT A (BIT)
PB <sub>7</sub> -PB <sub>0</sub>	PORT B (BIT)
PC <sub>7</sub> -PC <sub>0</sub>	PORT C (BIT)
V <sub>CC</sub>	+5 VOLTS
GND	0 VOLTS

### 8255A BLOCK DIAGRAM



## 8255A FUNCTIONAL DESCRIPTION

### General

The 8255A is a programmable peripheral interface (PPI) device designed for use in Intel® microcomputer systems. Its function is that of a general purpose I/O component to interface peripheral equipment to the microcomputer system bus. The functional configuration of the 8255A is programmed by the system software so that normally no external logic is necessary to interface peripheral devices or structures.

### Data Bus Buffer

This 3-state bidirectional 8-bit buffer is used to interface the 8255A to the system data bus. Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU. Control words and status information are also transferred through the data bus buffer.

### Read/Write and Control Logic

The function of this block is to manage all of the internal and external transfers of both Data and Control or Status words. It accepts inputs from the CPU Address and Control busses and in turn, issues commands to both of the Control Groups.

### (CS)

**Chip Select.** A "low" on this input pin enables the communication between the 8255A and the CPU.

### (RD)

**Read.** A "low" on this input pin enables the 8255A to send the data or status information to the CPU on the data bus. In essence, it allows the CPU to "read from" the 8255A.

### (WR)

**Write.** A "low" on this input pin enables the CPU to write data or control words into the 8255A.

### (A<sub>0</sub> and A<sub>1</sub>)

**Port Select 0 and Port Select 1.** These input signals, in conjunction with the RD and WR inputs, control the selection of one of the three ports or the control word registers. They are normally connected to the least significant bits of the address bus (A<sub>0</sub> and A<sub>1</sub>).

## 8255A BASIC OPERATION

A <sub>1</sub>	A <sub>0</sub>	RD	WR	CS	INPUT OPERATION (READ)
0	0	0	1	0	PORT A ⇒ DATA BUS
0	1	0	1	0	PORT B ⇒ DATA BUS
1	0	0	1	0	PORT C ⇒ DATA BUS
					OUTPUT OPERATION (WRITE)
0	0	1	0	0	DATA BUS ⇒ PORT A
0	1	1	0	0	DATA BUS ⇒ PORT B
1	0	1	0	0	DATA BUS ⇒ PORT C
1	1	1	0	0	DATA BUS ⇒ CONTROL
					DISABLE FUNCTION
X	X	X	X	1	DATA BUS ⇒ 3-STATE
1	1	0	1	0	ILLEGAL CONDITION
X	X	1	1	0	DATA BUS ⇒ 3-STATE

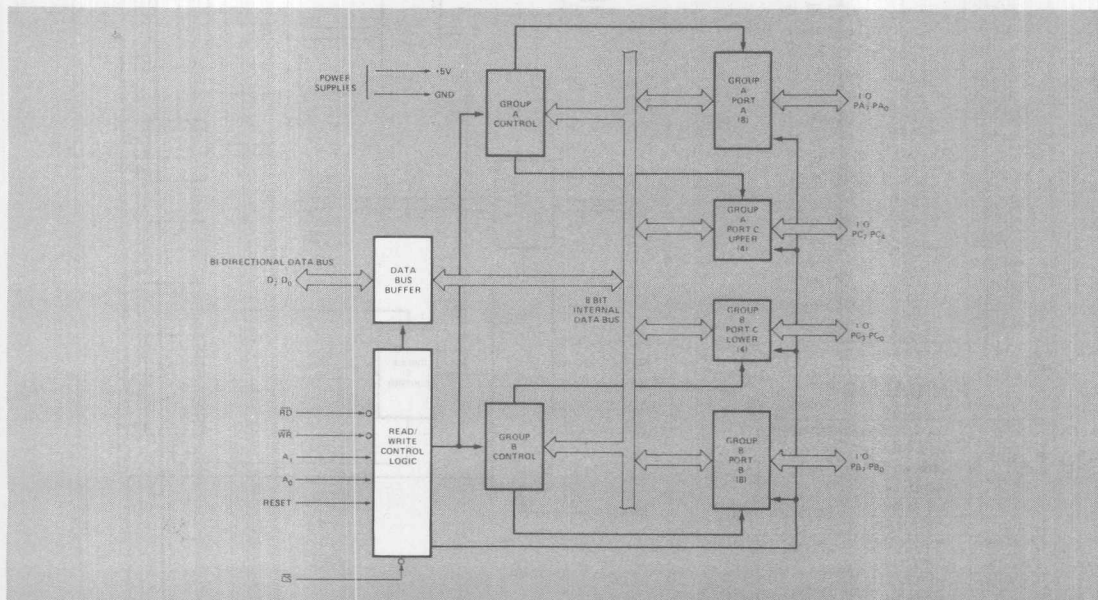


Figure 1. 8255A Block Diagram Showing Data Bus Buffer and Read/Write Control Logic Functions

**(RESET)**

**Reset.** A "high on this input clears the control register and all ports (A, C, C) are set to the input mode.

**Group A and Group B Controls**

The functional configuration of each port is programmed by the systems software. In essence, the CPU "outputs" a control word to the 8255A. The control word contains information such as "mode", "bit set", "bit reset", etc., that initializes the functional configuration of the 8255.

Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control Logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.

Control Group A — Port A and Port C upper (C7-C4)

Control Group B — Port B and Port C lower (C3-C0)

The Control Word Register can **Only** be written into. No Read operation of the Control Word Register is allowed.

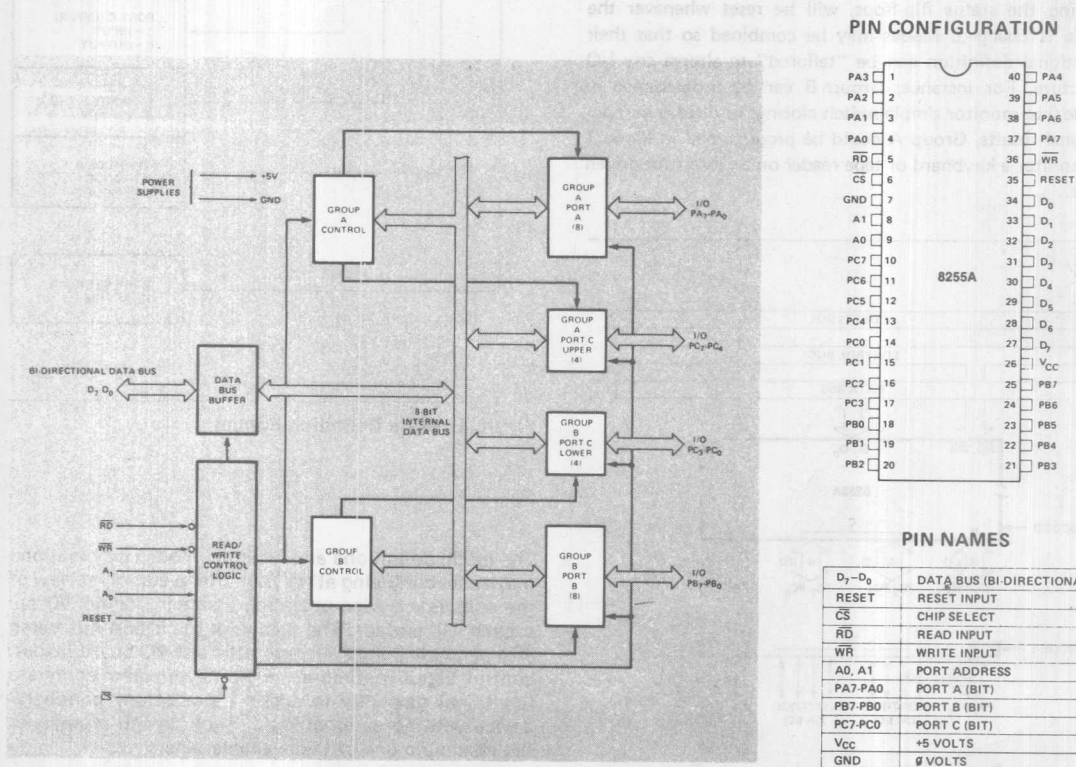
**Ports A, B, and C**

The 8255A contains three 8-bit ports (A, B, and C). All can be configured in a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 8255A.

**Port A.** One 8-bit data output latch/buffer and one 8-bit data input latch.

**Port B.** One 8-bit data input/output latch/buffer and one 8-bit data input buffer.

**Port C.** One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B.



**Figure 2. 8255A Block Diagram Showing Group A and Group B Control Functions**

## 8255A OPERATIONAL DESCRIPTION

### Mode Selection

There are three basic modes of operation that can be selected by the system software:

- Mode 0 — Basic Input/Output
- Mode 1 — Strobed Input/Output
- Mode 2 — Bi-Directional Bus

When the reset input goes "high" all ports will be set to the input mode (i.e., all 24 lines will be in the high impedance state). After the reset is removed the 8255A can remain in the input mode with no additional initialization required. During the execution of the system program any of the other modes may be selected using a single output instruction. This allows a single 8255A to service a variety of peripheral devices with a simple software maintenance routine.

The modes for Port A and Port B can be separately defined, while Port C is divided into two portions as required by the Port A and Port B definitions. All of the output registers, including the status flip-flops, will be reset whenever the mode is changed. Modes may be combined so that their functional definition can be "tailored" to almost any I/O structure. For instance; Group B can be programmed in Mode 0 to monitor simple switch closings or display computational results, Group A could be programmed in Mode 1 to monitor a keyboard or tape reader on an interrupt-driven basis.

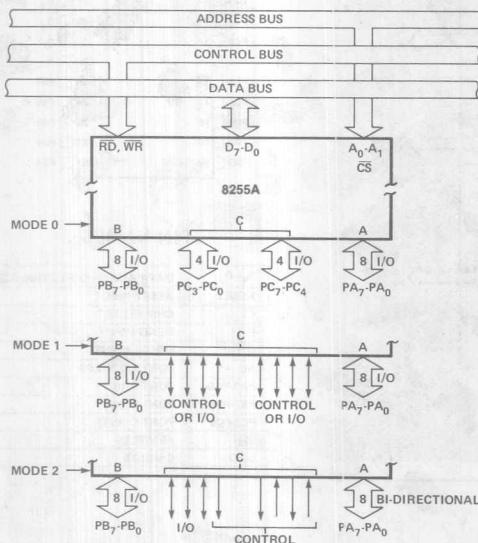


Figure 3. Basic Mode Definitions and Bus Interface

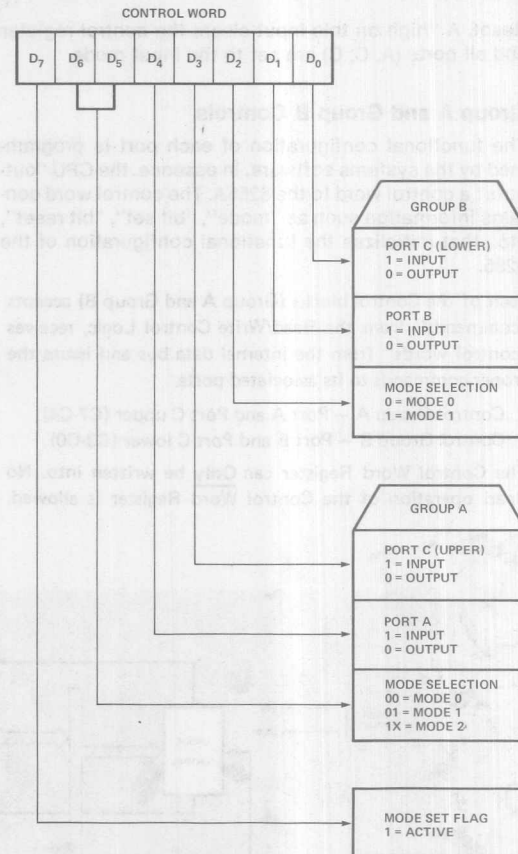


Figure 4. Mode Definition Format

The mode definitions and possible mode combinations may seem confusing at first but after a cursory review of the complete device operation a simple, logical I/O approach will surface. The design of the 8255A has taken into account things such as efficient PC board layout, control signal definition vs PC layout and complete functional flexibility to support almost any peripheral device with no external logic. Such design represents the maximum use of the available pins.

### Single Bit Set/Reset Feature

Any of the eight bits of Port C can be Set or Reset using a single OUTPUT instruction. This feature reduces software requirements in Control-based applications.

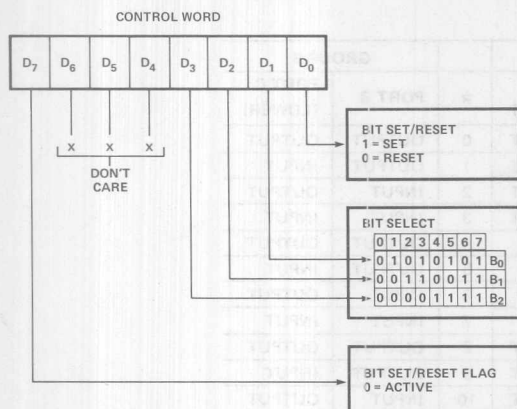


Figure 5. Bit Set/Reset Format

### Operating Modes

**MODE 0 (Basic Input/Output).** This functional configuration provides simple input and output operations for each of the three ports. No "handshaking" is required, data is simply written to or read from a specified port.

When Port C is being used as status/control for Port A or B, these bits can be set or reset by using the Bit Set/Reset operation just as if they were data output ports.

### Interrupt Control Functions

When the 8255A is programmed to operate in mode 1 or mode 2, control signals are provided that can be used as interrupt request inputs to the CPU. The interrupt request signals, generated from port C, can be inhibited or enabled by setting or resetting the associated INTE flip-flop, using the bit set/reset function of port C.

This function allows the Programmer to disallow or allow a specific I/O device to interrupt the CPU without affecting any other device in the interrupt structure.

INTE flip-flop definition:

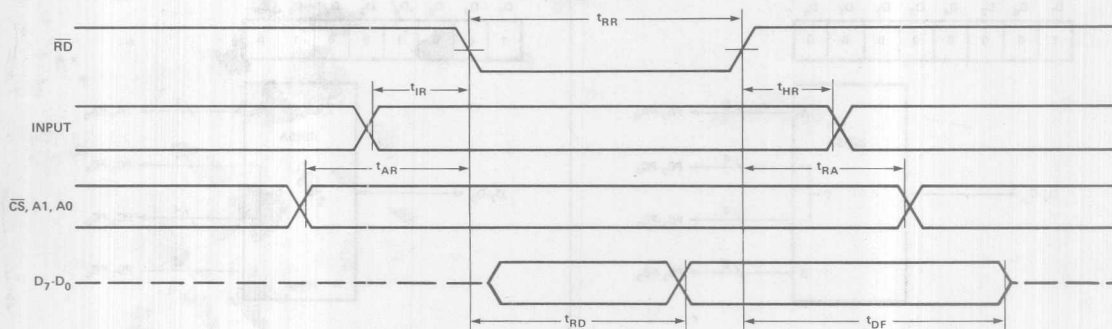
(BIT-SET) — INTE is SET — Interrupt enable

(BIT-RESET) — INTE is RESET — Interrupt disable

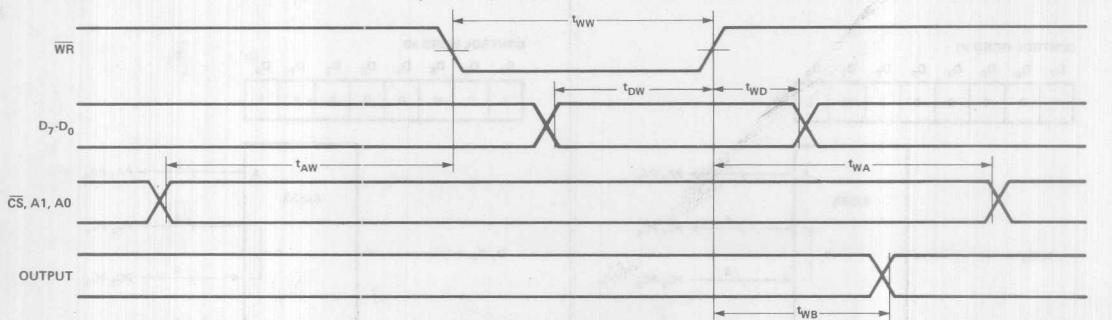
Note: All Mask flip-flops are automatically reset during mode selection and device Reset.

Mode 0 Basic Functional Definitions:

- Two 8-bit ports and two 4-bit ports.
- Any port can be input or output.
- Outputs are latched.
- Inputs are not latched.
- 16 different Input/Output configurations are possible in this Mode.



### MODE 0 (Basic Input)



### MODE 0 (Basic Output)

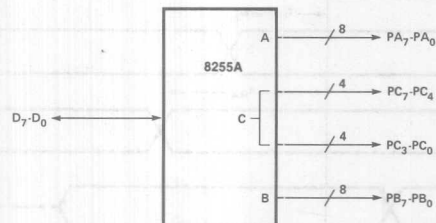
## MODE 0 Port Definition

A		B		GROUP A		GROUP B		
D <sub>4</sub>	D <sub>3</sub>	D <sub>1</sub>	D <sub>0</sub>	PORT A	PORT C (UPPER)	#	PORT B	PORT C (LOWER)
0	0	0	0	OUTPUT	OUTPUT	0	OUTPUT	OUTPUT
0	0	0	1	OUTPUT	OUTPUT	1	OUTPUT	INPUT
0	0	1	0	OUTPUT	OUTPUT	2	INPUT	OUTPUT
0	0	1	1	OUTPUT	OUTPUT	3	INPUT	INPUT
0	1	0	0	OUTPUT	INPUT	4	OUTPUT	OUTPUT
0	1	0	1	OUTPUT	INPUT	5	OUTPUT	INPUT
0	1	1	0	OUTPUT	INPUT	6	INPUT	OUTPUT
0	1	1	1	OUTPUT	INPUT	7	INPUT	INPUT
1	0	0	0	INPUT	OUTPUT	8	OUTPUT	OUTPUT
1	0	0	1	INPUT	OUTPUT	9	OUTPUT	INPUT
1	0	1	0	INPUT	OUTPUT	10	INPUT	OUTPUT
1	0	1	1	INPUT	OUTPUT	11	INPUT	INPUT
1	1	0	0	INPUT	INPUT	12	OUTPUT	OUTPUT
1	1	0	1	INPUT	INPUT	13	OUTPUT	INPUT
1	1	1	0	INPUT	INPUT	14	INPUT	OUTPUT
1	1	1	1	INPUT	INPUT	15	INPUT	INPUT

## MODE 0 Configurations

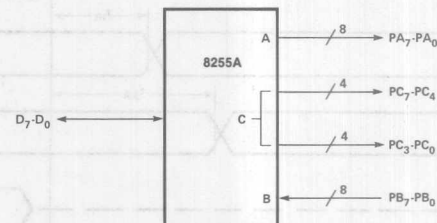
CONTROL WORD #0

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
1	0	0	0	0	0	0	0



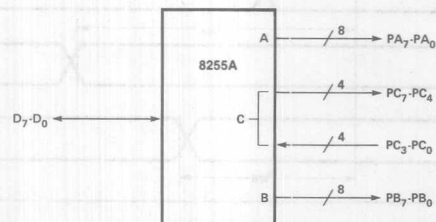
CONTROL WORD #2

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
1	0	0	0	0	0	1	0



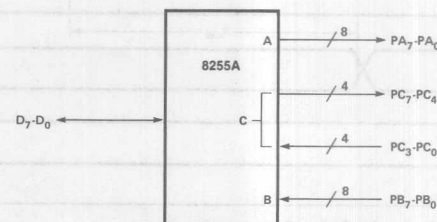
CONTROL WORD #1

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
1	0	0	0	0	0	0	1



CONTROL WORD #3

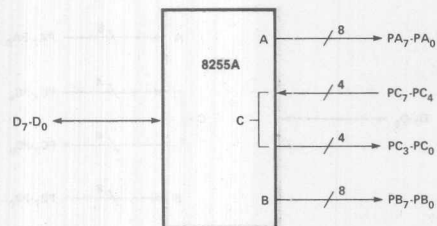
D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
1	0	0	0	0	0	1	1



# 8255A/8255A-5

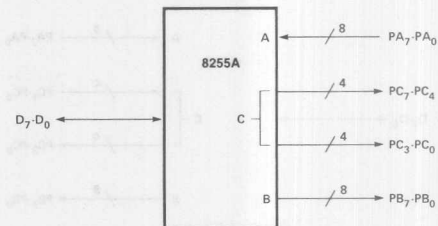
CONTROL WORD #4

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
1	0	0	0	1	0	0	0



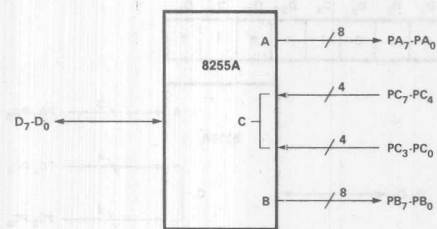
CONTROL WORD #8

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
1	0	0	1	0	0	0	0



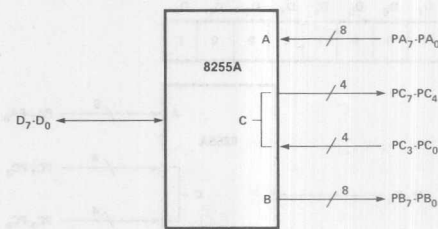
CONTROL WORD #5

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
1	0	0	0	1	0	0	1



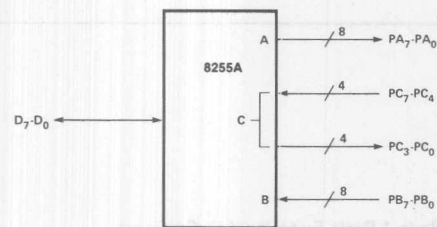
CONTROL WORD #9

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
1	0	0	1	0	0	0	1



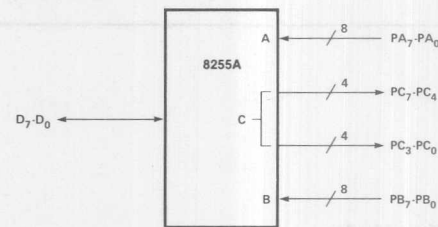
CONTROL WORD #6

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
1	0	0	0	1	0	1	0



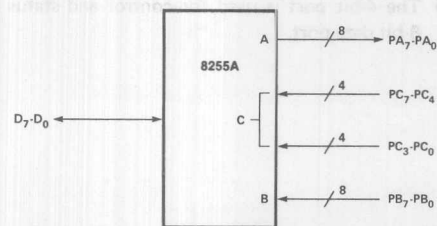
CONTROL WORD #10

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
1	0	0	1	0	0	1	0



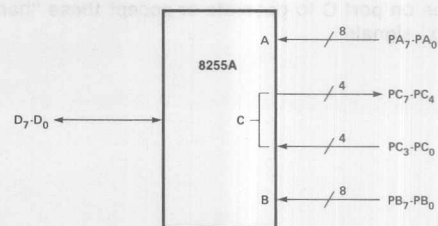
CONTROL WORD #7

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
1	0	0	0	1	0	1	1



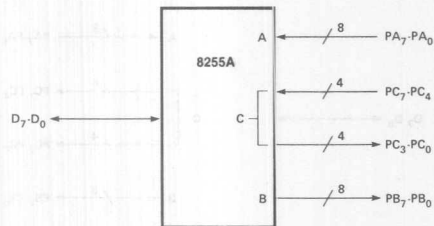
CONTROL WORD #11

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
1	0	0	1	0	0	1	1



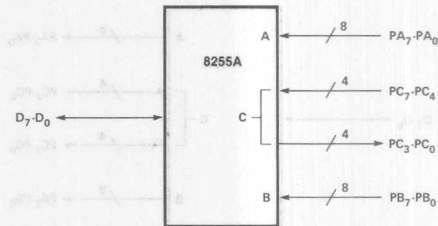
CONTROL WORD #12

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
1	0	0	1	1	0	0	0



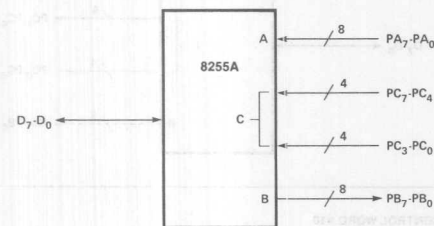
CONTROL WORD #14

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
1	0	0	1	1	0	1	0



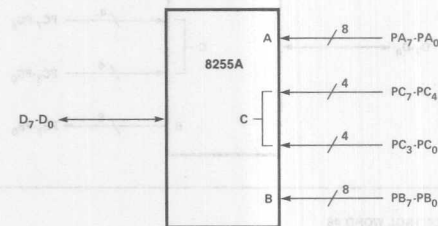
CONTROL WORD #13

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
1	0	0	1	1	0	0	1



CONTROL WORD #15

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
1	0	0	1	1	0	1	1



## Operating Modes

**MODE 1 (Strobed Input/Output).** This functional configuration provides a means for transferring I/O data to or from a specified port in conjunction with strobes or "handshaking" signals. In mode 1, port A and Port B use the lines on port C to generate or accept these "handshaking" signals.

### Mode 1 Basic Functional Definitions:

- Two Groups (Group A and Group B)
- Each group contains one 8-bit data port and one 4-bit control/data port.
- The 8-bit data port can be either input or output. Both inputs and outputs are latched.
- The 4-bit port is used for control and status of the 8-bit data port.

### Input Control Signal Definition

**STB (Strobe Input).** A "low" on this input loads data into the input latch.

### IBF (Input Buffer Full F/F)

A "high" on this output indicates that the data has been loaded into the input latch; in essence, an acknowledgement. IBF is set by STB input being low and is reset by the rising edge of the RD input.

### INTR (Interrupt Request)

A "high" on this output can be used to interrupt the CPU when an input device is requesting service. INTR is set by the STB is a "one", IBF is a "one" and INTE is a "one". It is reset by the falling edge of RD. This procedure allows an input device to request service from the CPU by simply strobing its data into the port.

#### INTE A

Controlled by bit set/reset of PC<sub>4</sub>.

#### INTE B

Controlled by bit set/reset of PC<sub>2</sub>.

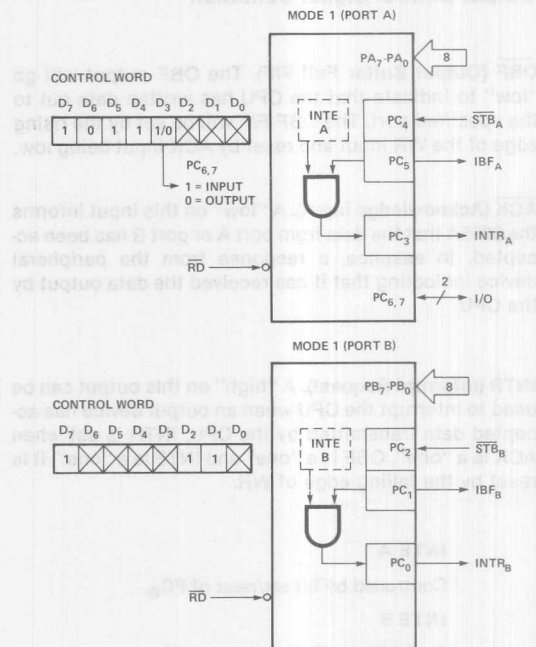


Figure 6. MODE 1 Input

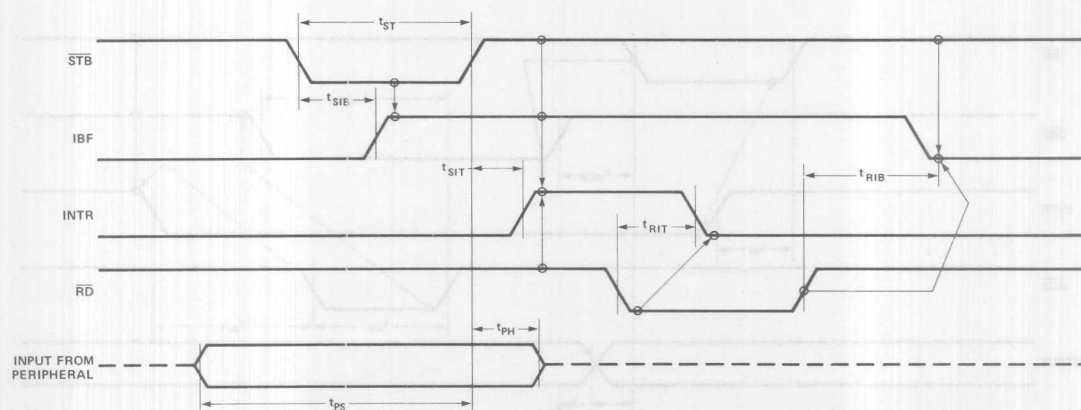


Figure 7. MODE 1 (Strobed Input)

## Output Control Signal Definition

**OBF (Output Buffer Full F/F).** The OBF output will go "low" to indicate that the CPU has written data out to the specified port. The OBF F/F will be set by the rising edge of the WR input and reset by ACK Input being low.

**ACK (Acknowledge Input).** A "low" on this input informs the 8255A that the data from port A or port B has been accepted. In essence, a response from the peripheral device indicating that it has received the data output by the CPU.

**INTR (Interrupt Request).** A "high" on this output can be used to interrupt the CPU when an output device has accepted data transmitted by the CPU. INTR is set when ACK is a "one", OBF is a "one" and INTE is a "one". It is reset by the falling edge of WR.

### INTE A

Controlled by bit set/reset of PC<sub>6</sub>.

### INTE B

Controlled by bit set/reset of PC<sub>2</sub>.

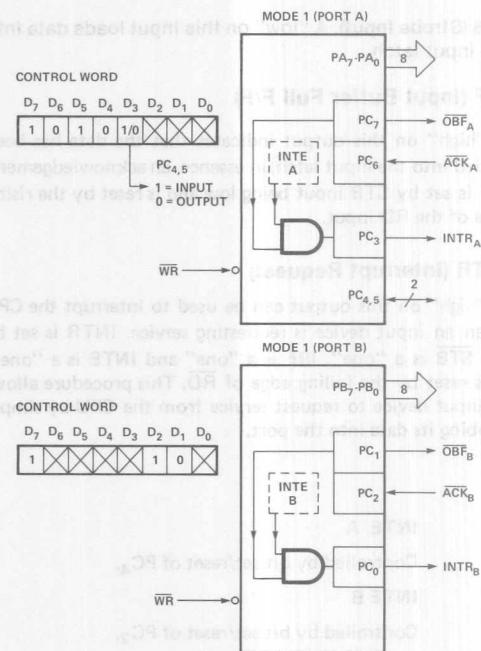


Figure 8. MODE 1 Output

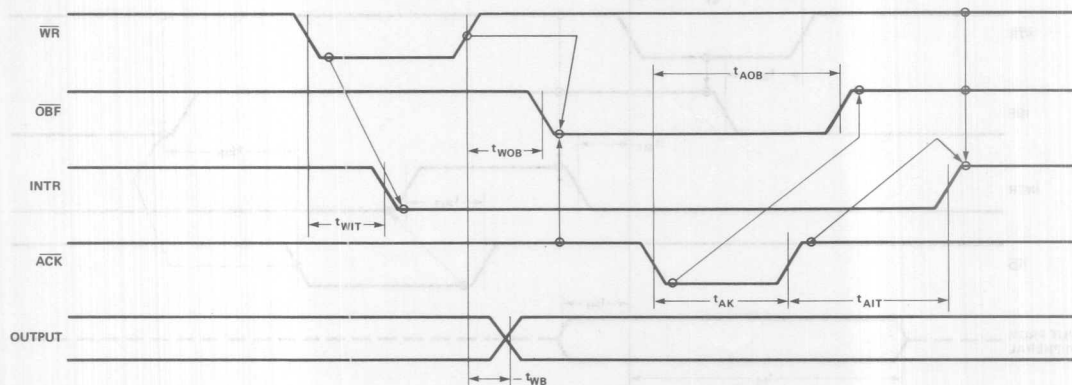


Figure 9. Mode 1 (Strobed Output)

### Combinations of MODE 1

Port A and Port B can be individually defined as input or output in Mode 1 to support a wide variety of strobed I/O applications.

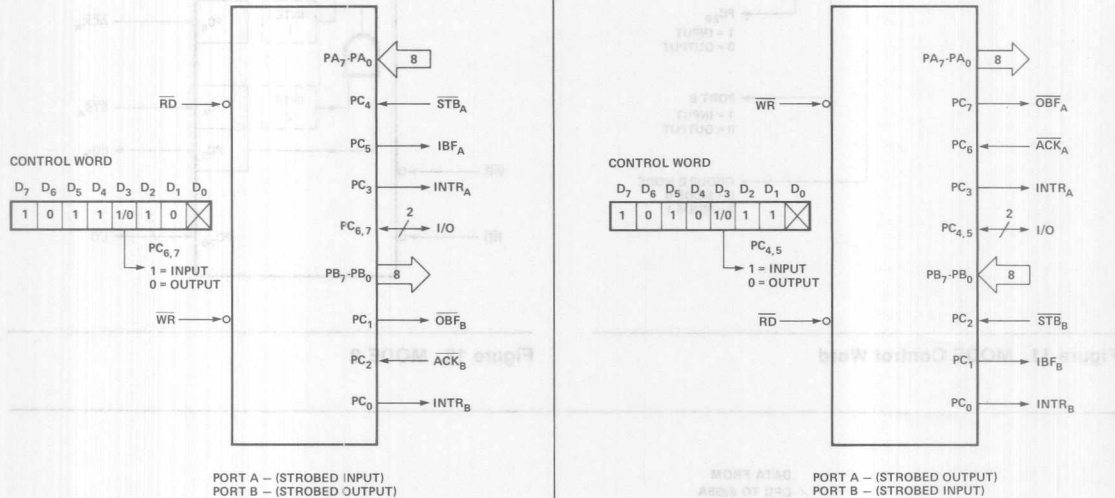


Figure 10. Combinations of MODE 1

### Operating Modes

**MODE 2 (Strobed Bidirectional Bus I/O).** This functional configuration provides a means for communicating with a peripheral device or structure on a single 8-bit bus for both transmitting and receiving data (bidirectional bus I/O). "Handshaking" signals are provided to maintain proper bus flow discipline in a similar manner to MODE 1. Interrupt generation and enable/disable functions are also available.

#### MODE 2 Basic Functional Definitions:

- Used in Group A only.
- One 8-bit, bi-directional bus Port (Port A) and a 5-bit control Port (Port C).
- Both inputs and outputs are latched.
- The 5-bit control port (Port C) is used for control and status for the 8-bit, bi-directional bus port (Port A).

### Bidirectional Bus I/O Control Signal Definition

**INTR (Interrupt Request).** A high on this output can be used to interrupt the CPU for both input or output operations.

### Output Operations

**OBF (Output Buffer Full).** The OBF output will go "low" to indicate that the CPU has written data out to port A.

**ACK (Acknowledge).** A "low" on this input enables the tri-state output buffer of port A to send out the data. Otherwise, the output buffer will be in the high impedance state.

**INTE 1 (The INTE Flip-Flop Associated with OBF).** Controlled by bit set/reset of PC<sub>6</sub>.

### Input Operations

**STB (Strobe Input)**

**STB (Strobe Input).** A "low" on this input loads data into the input latch.

**IBF (Input Buffer Full F/F).** A "high" on this output indicates that data has been loaded into the input latch.

**INTE 2 (The INTE Flip-Flop Associated with IBF).** Controlled by bit set/reset of PC<sub>4</sub>.

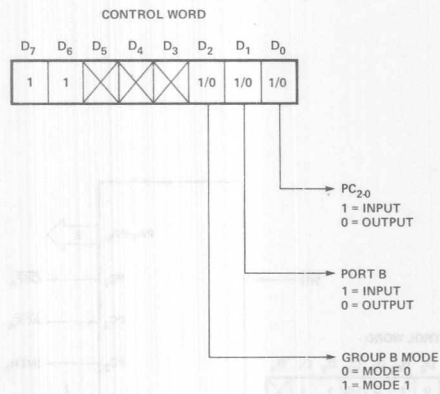


Figure 11. MODE Control Word

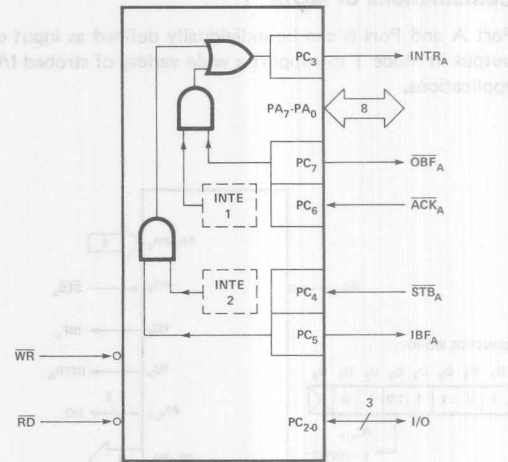


Figure 12. MODE 2

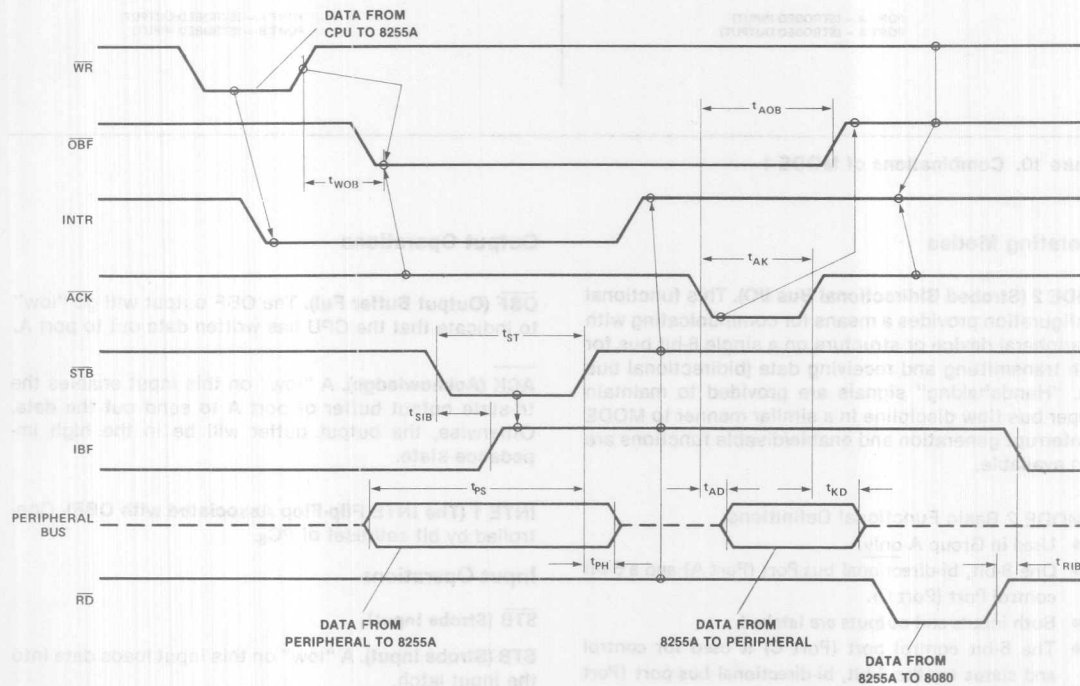


Figure 13. MODE 2 (Bidirectional)

NOTE: Any sequence where  $\overline{WR}$  occurs before  $\overline{ACK}$  and  $\overline{STB}$  occurs before  $\overline{RD}$  is permissible.  
( $INTR = IBF \cdot MASK \cdot STB \cdot RD + OBF \cdot MASK \cdot ACK \cdot WR$ )

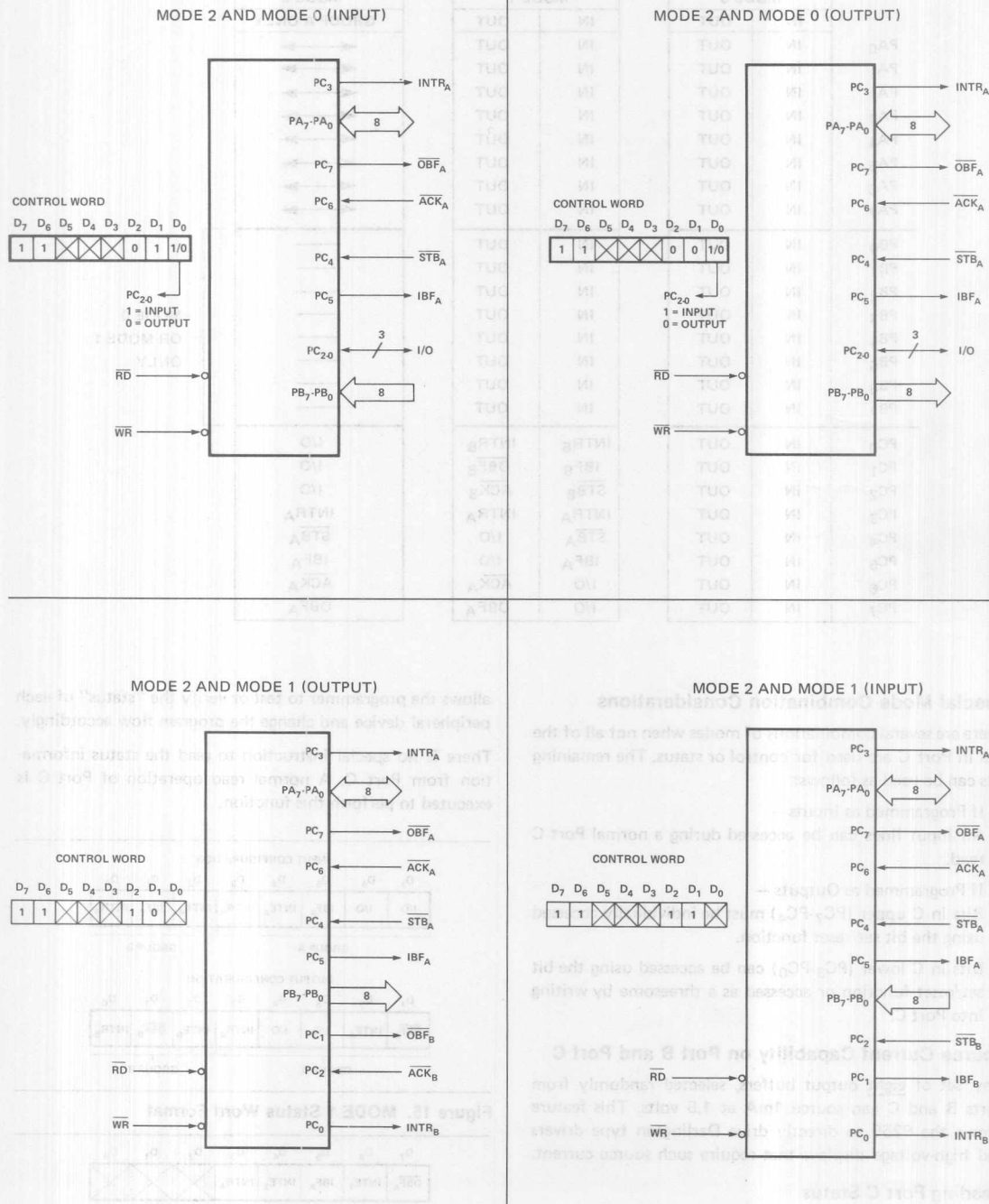


Figure 14. MODE 2 Combinations

## Mode Definition Summary

	MODE 0		MODE 1		MODE 2	
	IN	OUT	IN	OUT	GROUP A ONLY	
PA <sub>0</sub>	IN	OUT	IN	OUT	↔	
PA <sub>1</sub>	IN	OUT	IN	OUT	↔	
PA <sub>2</sub>	IN	OUT	IN	OUT	↔	
PA <sub>3</sub>	IN	OUT	IN	OUT	↔	
PA <sub>4</sub>	IN	OUT	IN	OUT	↔	
PA <sub>5</sub>	IN	OUT	IN	OUT	↔	
PA <sub>6</sub>	IN	OUT	IN	OUT	↔	
PA <sub>7</sub>	IN	OUT	IN	OUT	↔	
PB <sub>0</sub>	IN	OUT	IN	OUT	—	
PB <sub>1</sub>	IN	OUT	IN	OUT	—	
PB <sub>2</sub>	IN	OUT	IN	OUT	—	
PB <sub>3</sub>	IN	OUT	IN	OUT	—	
PB <sub>4</sub>	IN	OUT	IN	OUT	—	
PB <sub>5</sub>	IN	OUT	IN	OUT	—	
PB <sub>6</sub>	IN	OUT	IN	OUT	—	
PB <sub>7</sub>	IN	OUT	IN	OUT	—	
PC <sub>0</sub>	IN	OUT	INTR <sub>B</sub>	INTR <sub>B</sub>	I/O	
PC <sub>1</sub>	IN	OUT	IBF <sub>B</sub>	OBFB	I/O	
PC <sub>2</sub>	IN	OUT	STB <sub>B</sub>	ACK <sub>B</sub>	I/O	
PC <sub>3</sub>	IN	OUT	INTR <sub>A</sub>	INTR <sub>A</sub>	INTR <sub>A</sub>	
PC <sub>4</sub>	IN	OUT	STB <sub>A</sub>	I/O	STB <sub>A</sub>	
PC <sub>5</sub>	IN	OUT	IBF <sub>A</sub>	I/O	IBF <sub>A</sub>	
PC <sub>6</sub>	IN	OUT	I/O	ACK <sub>A</sub>	ACK <sub>A</sub>	
PC <sub>7</sub>	IN	OUT	I/O	OBFA	OBFA	

MODE 0  
OR MODE 1  
ONLY

## Special Mode Combination Considerations

There are several combinations of modes when not all of the bits in Port C are used for control or status. The remaining bits can be used as follows:

If Programmed as Inputs —

All input lines can be accessed during a normal Port C read.

If Programmed as Outputs —

Bits in C upper (PC<sub>7</sub>-PC<sub>4</sub>) must be individually accessed using the bit set/reset function.

Bits in C lower (PC<sub>3</sub>-PC<sub>0</sub>) can be accessed using the bit set/reset function or accessed as a threesome by writing into Port C.

## Source Current Capability on Port B and Port C

Any set of **eight** output buffers, selected randomly from Ports B and C can source 1mA at 1.5 volts. This feature allows the 8255 to directly drive Darlington type drivers and high-voltage displays that require such source current.

## Reading Port C Status

In Mode 0, Port C transfers data to or from the peripheral device. When the 8255 is programmed to function in Modes 1 or 2, Port C generates or accepts "hand-shaking" signals with the peripheral device. Reading the contents of Port C

allows the programmer to test or verify the "status" of each peripheral device and change the program flow accordingly.

There is no special instruction to read the status information from Port C. A normal read operation of Port C is executed to perform this function.

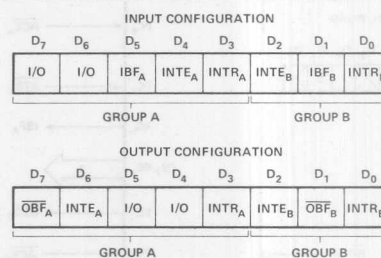


Figure 15. MODE 1 Status Word Format

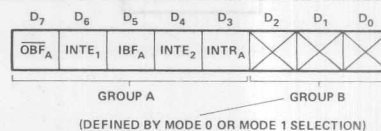


Figure 16. MODE 2 Status Word Format

## APPLICATIONS OF THE 8255A

The 8255A is a very powerful tool for interfacing peripheral equipment to the microcomputer system. It represents the optimum use of available pins and is flexible enough to interface almost any I/O device without the need for additional external logic.

Each peripheral device in a microcomputer system usually has a "service routine" associated with it. The routine manages the software interface between the device and the CPU. The functional definition of the 8255A is programmed by the I/O service routine and becomes an extension of the system software. By examining the I/O devices interface characteristics for both data transfer and timing, and matching this information to the examples and tables in the detailed operational description, a control word can easily be developed to initialize the 8255A to exactly "fit" the application. Figures 17 through 23 present a few examples of typical applications of the 8255A.

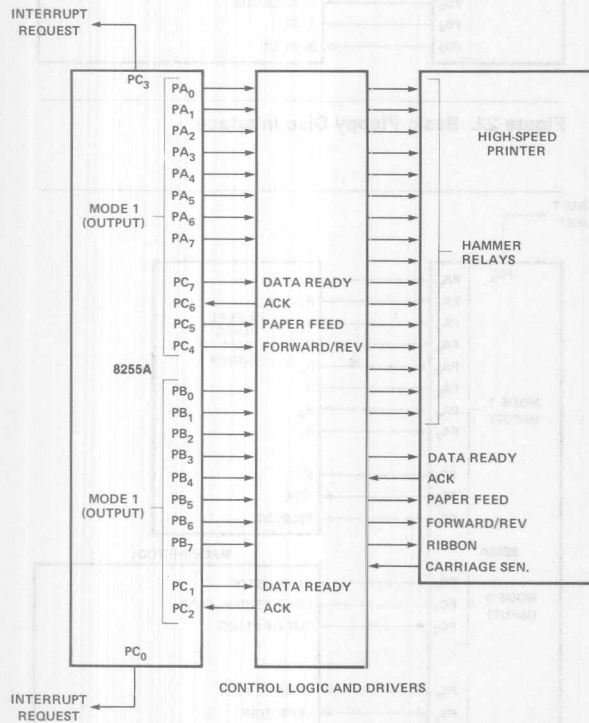


Figure 17. Printer Interface

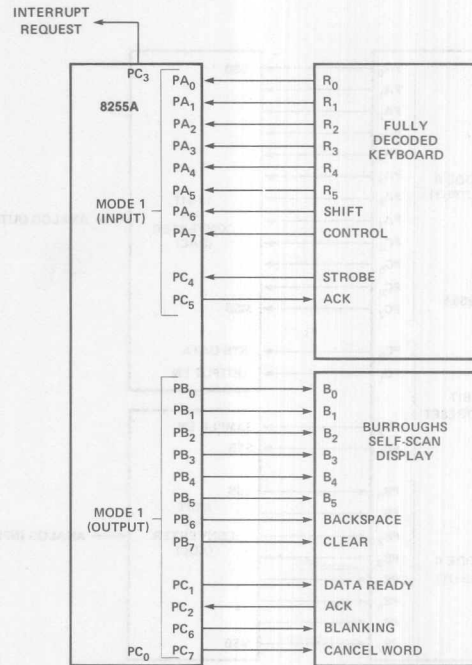


Figure 18. Keyboard and Display Interface

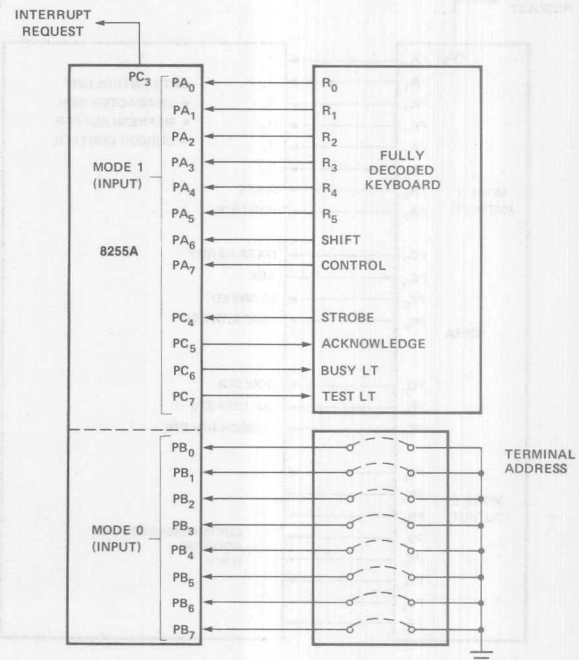


Figure 19. Keyboard and Terminal Address Interface

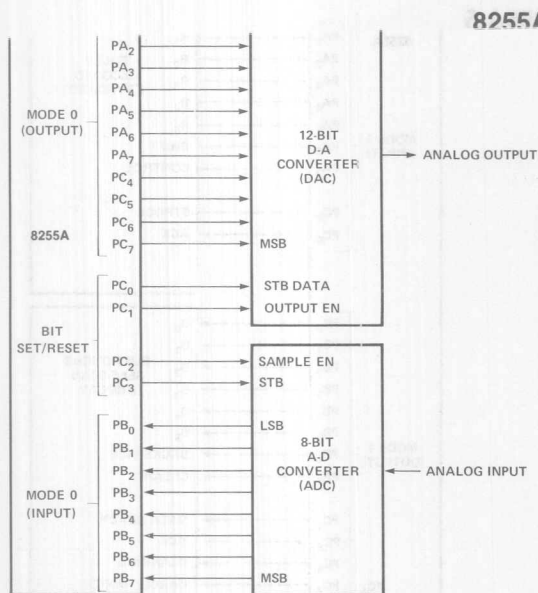


Figure 20. Digital to Analog, Analog to Digital

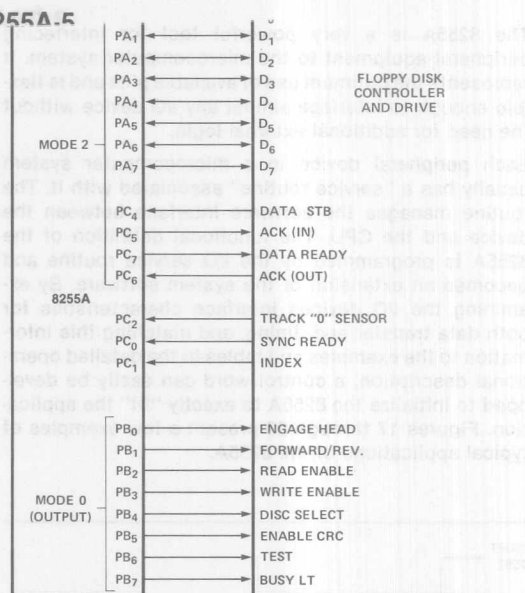


Figure 22. Basic Floppy Disc Interface

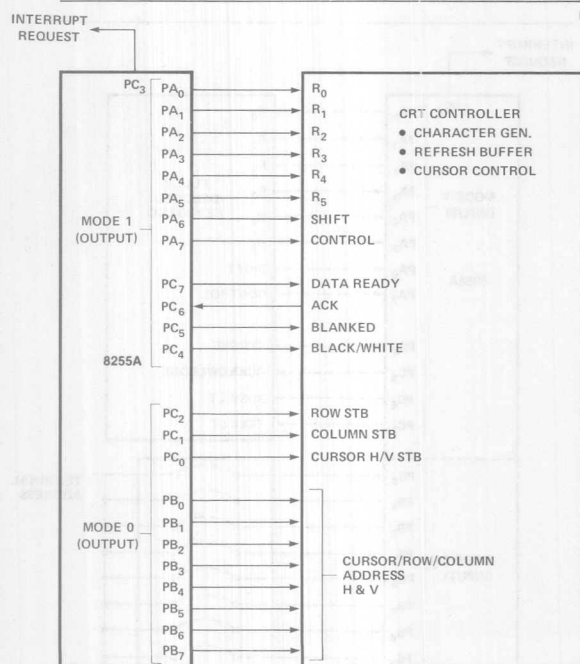


Figure 21. Basic CRT Controller Interface

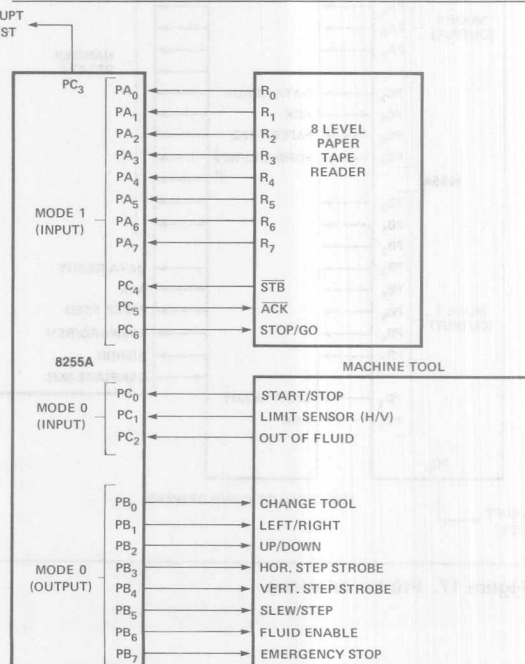


Figure 23. Machine Tool Controller Interface

**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature Under Bias. . . . .  $0^{\circ}\text{C}$  to  $70^{\circ}\text{C}$   
 Storage Temperature . . . . .  $-65^{\circ}\text{C}$  to  $+150^{\circ}\text{C}$   
 Voltage on Any Pin  
     With Respect to Ground. . . . .  $-0.5\text{V}$  to  $+7\text{V}$   
 Power Dissipation . . . . . 1 Watt

*\*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

**D.C. CHARACTERISTICS**

$T_A = 0^{\circ}\text{C}$  to  $70^{\circ}\text{C}$ ,  $V_{CC} = +5\text{V} \pm 5\%$ ;  $\text{GND} = 0\text{V}$

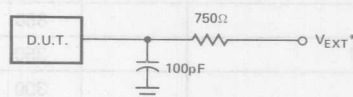
SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
$V_{IL}$	Input Low Voltage	-0.5	0.8	V	
$V_{IH}$	Input High Voltage	2.0	$V_{CC}$	V	
$V_{OL}(\text{DB})$	Output Low Voltage (Data Bus)		0.45	V	$I_{OL} = 2.5\text{mA}$
$V_{OL}(\text{PER})$	Output Low Voltage (Peripheral Port)		0.45	V	$I_{OL} = 1.7\text{mA}$
$V_{OH}(\text{DB})$	Output High Voltage (Data Bus)	2.4		V	$I_{OH} = -400\mu\text{A}$
$V_{OH}(\text{PER})$	Output High Voltage (Peripheral Port)	2.4		V	$I_{OH} = -200\mu\text{A}$
$I_{DAR}^{[1]}$	Darlington Drive Current	-1.0	-4.0	mA	$R_{EXT} = 750\Omega$ ; $V_{EXT} = 1.5\text{V}$
$I_{CC}$	Power Supply Current		120	mA	
$I_{IL}$	Input Load Current		$\pm 10$	$\mu\text{A}$	$V_{IN} = V_{CC}$ to $0\text{V}$
$I_{OFL}$	Output Float Leakage		$\pm 10$	$\mu\text{A}$	$V_{OUT} = V_{CC}$ to $0\text{V}$

Note 1: Available on any 8 pins from Port B and C.

**CAPACITANCE**

$T_A = 25^{\circ}\text{C}$ ;  $V_{CC} = \text{GND} = 0\text{V}$

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNIT	TEST CONDITIONS
$C_{IN}$	Input Capacitance			10	pF	$f_c = 1\text{MHz}$
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured pins returned to GND



\* $V_{EXT}$  is set at various voltages during testing to guarantee the specification.

Figure 24. Test Load Circuit (for dB)

## A.C. CHARACTERISTICS

$T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ;  $V_{CC} = +5V \pm 5\%$ ;  $GND = 0V$

## Bus Parameters

## Read:

SYMBOL	PARAMETER	8255A		8255A-5		UNIT
		MIN.	MAX.	MIN.	MAX.	
$t_{AR}$	Address Stable Before READ	0		0		ns
$t_{RA}$	Address Stable After READ	0		0		ns
$t_{RR}$	READ Pulse Width	300		300		ns
$t_{RD}$	Data Valid From READ <sup>[1]</sup>		250		200	ns
$t_{DF}$	Data Float After READ	10	150	10	100	ns
$t_{RV}$	Time Between READs and/or WRITEs	850		850		ns

NOTE:  
The 8255A-5 specifications are not final. Some parametric limits are subject to change.

## Write:

SYMBOL	PARAMETER	8255A		8255A-5		UNIT
		MIN.	MAX.	MIN.	MAX.	
$t_{AW}$	Address Stable Before WRITE	0		0		ns
$t_{WA}$	Address Stable After WRITE	20		20		ns
$t_{WW}$	WRITE Pulse Width	400		300		ns
$t_{DW}$	Data Valid to WRITE (T.E.)	100		100		ns
$t_{WD}$	Data Valid After WRITE	30		30		ns

## Other Timings:

SYMBOL	PARAMETER	8255A		8255A-5		UNIT
		MIN.	MAX.	MIN.	MAX.	
$t_{WB}$	WR = 1 to Output <sup>[1]</sup>		350		350	ns
$t_{IR}$	Peripheral Data Before RD	0		0		ns
$t_{HR}$	Peripheral Data After RD	0		0		ns
$t_{AK}$	ACK Pulse Width	300		300		ns
$t_{ST}$	STB Pulse Width	500		500		ns
$t_{PS}$	Per. Data Before T.E. of STB	0		0		ns
$t_{PH}$	Per. Data After T.E. of STB	180		180		ns
$t_{AD}$	ACK = 0 to Output <sup>[1]</sup>		300		300	ns
$t_{KD}$	ACK = 1 to Output Float	20	250	20	250	ns
$t_{WOB}$	WR = 1 to OBF = 0 <sup>[1]</sup>		650		650	ns
$t_{AOB}$	ACK = 0 to OBF = 1 <sup>[1]</sup>		350		350	ns
$t_{SIB}$	STB = 0 to IBF = 1 <sup>[1]</sup>		300		300	ns
$t_{RIB}$	RD = 1 to IBF = 0 <sup>[1]</sup>		300		300	ns
$t_{RIT}$	RD = 0 to INTR = 0 <sup>[1]</sup>		400		400	ns
$t_{SIT}$	STB = 1 to INTR = 1 <sup>[1]</sup>		300		300	ns
$t_{AIT}$	ACK = 1 to INTR = 1 <sup>[1]</sup>		350		350	ns
$t_{WIT}$	WR = 0 to INTR = 0 <sup>[1]</sup>		850		850	ns

- Notes: 1. Test Conditions: 8255A:  $C_L = 100\text{pF}$ ; 8255A-5:  $C_L = 150\text{pF}$ .  
2. Period of Reset pulse must be at least  $50\mu\text{s}$  during or after power on. Subsequent Reset pulse can be  $500\text{ ns min.}$

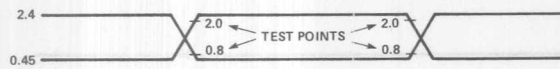


Figure 25. Input Waveforms for A.C. Tests

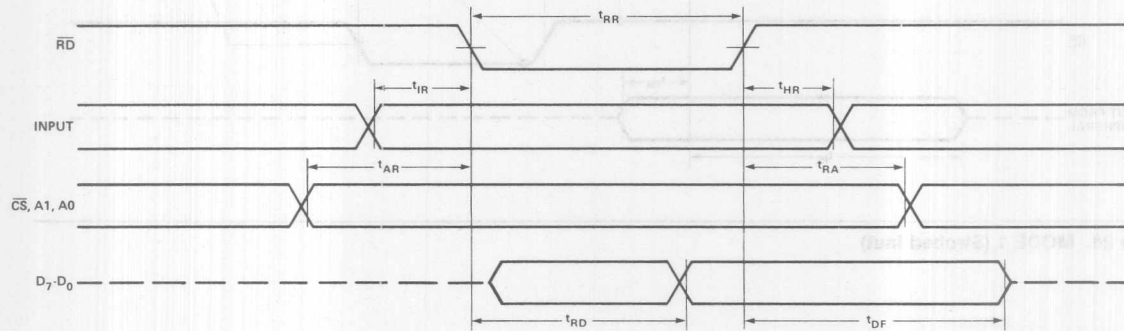


Figure 26. MODE 0 (Basic Input)

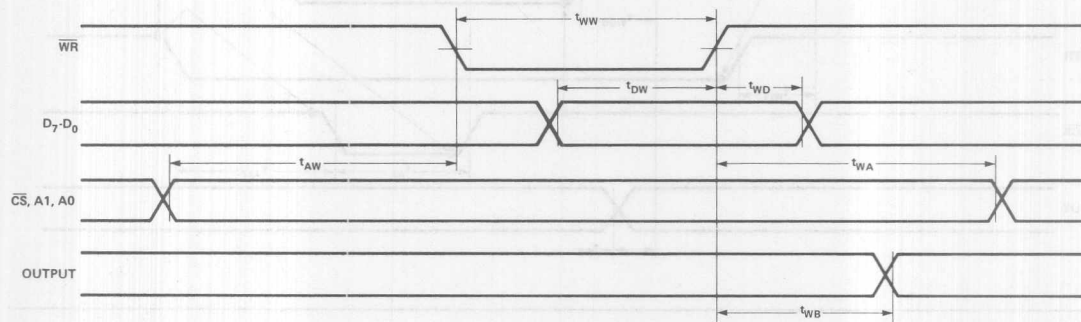


Figure 27. MODE 0 (Basic Output)

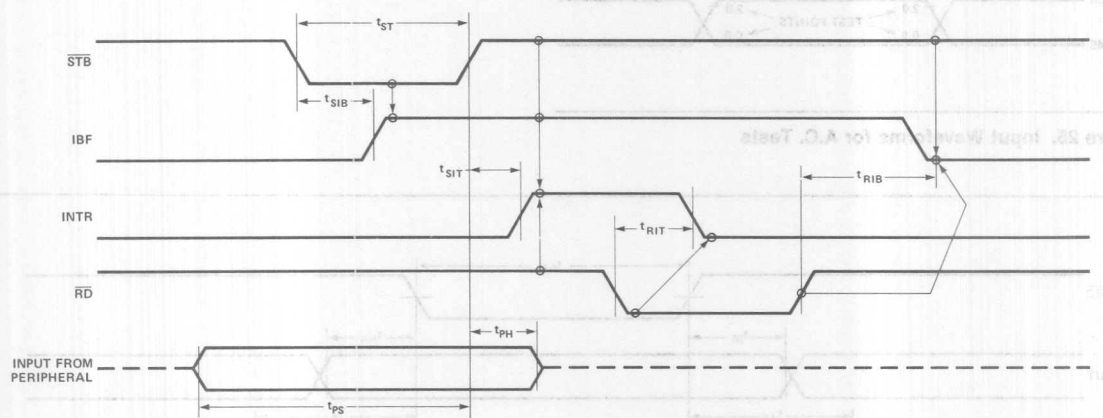


Figure 28. MODE 1 (Strobed Input)

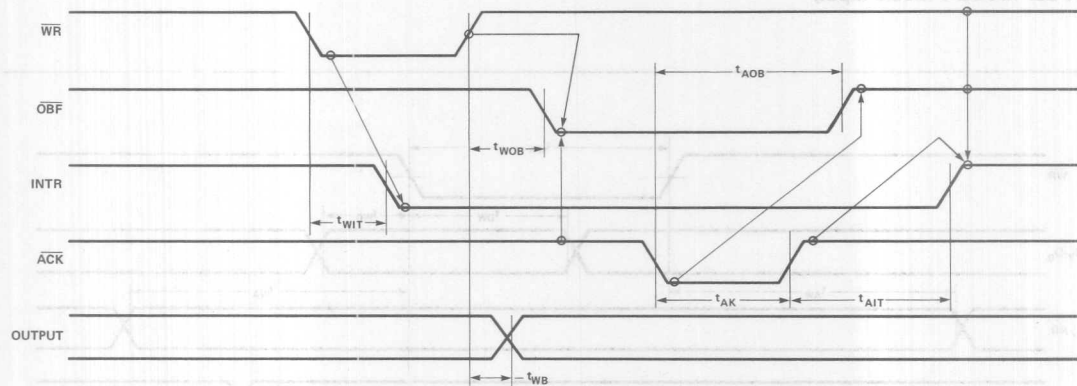


Figure 29. MODE 1 (Strobed Output)

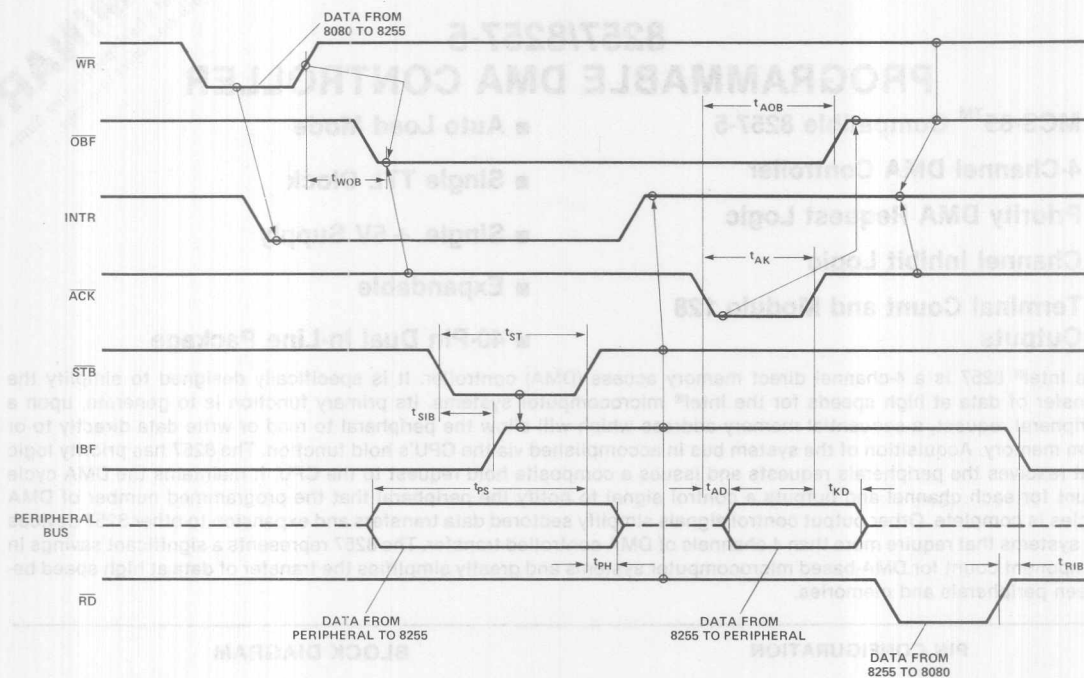


Figure 30. MODE 2 (Bidirectional)

NOTE: Any sequence where  $\overline{WR}$  occurs before  $\overline{ACK}$  and  $\overline{STB}$  occurs before  $\overline{RD}$  is permissible.  
 $(\overline{INTR} = \overline{IBF} \cdot \text{MASK} \cdot \overline{STB} \cdot \overline{RD} + \overline{OBF} \cdot \text{MASK} \cdot \overline{ACK} \cdot \overline{WR})$

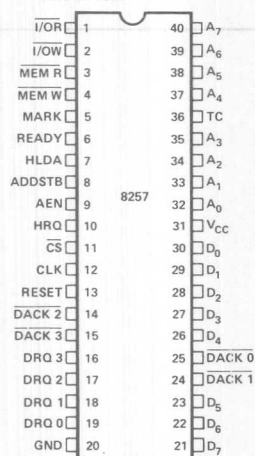
# 8257/8257-5

## PROGRAMMABLE DMA CONTROLLER

- MCS-85™ Compatible 8257-5
- 4-Channel DMA Controller
- Priority DMA Request Logic
- Channel Inhibit Logic
- Terminal Count and Modulo 128 Outputs
- Auto Load Mode
- Single TTL Clock
- Single +5V Supply
- Expandable
- 40-Pin Dual In-Line Package

The Intel® 8257 is a 4-channel direct memory access (DMA) controller. It is specifically designed to simplify the transfer of data at high speeds for the Intel® microcomputer systems. Its primary function is to generate, upon a peripheral request, a sequential memory address which will allow the peripheral to read or write data directly to or from memory. Acquisition of the system bus is accomplished via the CPU's hold function. The 8257 has priority logic that resolves the peripherals requests and issues a composite hold request to the CPU. It maintains the DMA cycle count for each channel and outputs a control signal to notify the peripheral that the programmed number of DMA cycles is complete. Other output control signals simplify sectorized data transfers and expansion to other 8257 devices for systems that require more than 4 channels of DMA controlled transfer. The 8257 represents a significant savings in component count for DMA-based microcomputer systems and greatly simplifies the transfer of data at high speed between peripherals and memories.

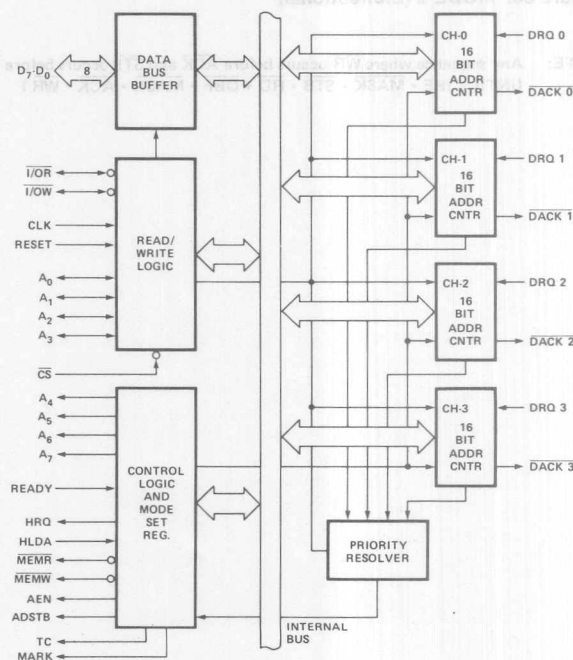
### PIN CONFIGURATION



### PIN NAMES

D <sub>7</sub> -D <sub>0</sub>	DATA BUS	AEN	ADDRESS ENABLE
A <sub>7</sub> -A <sub>0</sub>	ADDRESS BUS	ADSTB	ADDRESS STROBE
I/OR	I/O READ	TC	TERMINAL COUNT
I/OW	I/O WRITE	MARK	MODULO 128 MARK
MEMR	MEMORY READ	DRQ <sub>3</sub> -DRQ <sub>0</sub>	DMA REQUEST INPUT
MEMW	MEMORY WRITE	DACK <sub>3</sub> -DACK <sub>0</sub>	DMA ACKNOWLEDGE OUT
CLK	CLOCK INPUT	CS	CHIP SELECT
RESET	RESET INPUT	V <sub>CC</sub>	+5 VOLTS
READY	READY	GND	GROUND
HRQ	HOLD REQUEST (TO 8080A)		
HLDA	HOLD ACKNOWLEDGE (FROM 8080A)		

### BLOCK DIAGRAM



## FUNCTIONAL DESCRIPTION

### General

The 8257 is a programmable, Direct Memory Access (DMA) device which, when coupled with a single Intel® 8212 I/O port device, provides a complete four-channel DMA controller for use in Intel® microcomputer systems. After being initialized by software, the 8257 can transfer a block of data, containing up to 16,384 bytes, between memory and a peripheral device directly, without further intervention required of the CPU. Upon receiving a DMA transfer request from an enabled peripheral, the 8257:

- Acquires control of the system bus.
- Acknowledges that requesting peripheral which is connected to the highest priority channel.
- Outputs the least significant eight bits of the memory address onto system address lines A<sub>0</sub>-A<sub>7</sub>, outputs the most significant eight bits of the memory address to the 8212 I/O port via the data bus (the 8212 places these address bits on lines A<sub>8</sub>-A<sub>15</sub>), and
- Generates the appropriate memory and I/O read/write control signals that cause the peripheral to receive or deposit a data byte directly from or to the addressed location in memory.

The 8257 will retain control of the system bus and repeat the transfer sequence, as long as a peripheral maintains its DMA request. Thus, the 8257 can transfer a block of data to/from a high speed peripheral (e.g., a sector of data on a floppy disk) in a single "burst". When the specified number of data bytes have been transferred, the 8257 activates its Terminal Count (TC) output, informing the CPU that the operation is complete.

The 8257 offers three different modes of operation: (1) DMA read, which causes data to be transferred from memory to a peripheral; (2) DMA write, which causes data to be transferred from a peripheral to memory; and (3) DMA verify, which does not actually involve the transfer of data. When an 8257 channel is in the DMA verify mode, it will respond the same as described for transfer operations, except that no memory or I/O read/write control signals will be generated, thus preventing the transfer of data. The 8257, however, will gain control of the system bus and will acknowledge the peripheral's DMA request for each DMA cycle. The peripheral can use these acknowledge signals to enable an internal access of each byte of a data block in order to execute some verification procedure, such as the accumulation of a CRC (Cyclic Redundancy Code) checkword. For example, a block of DMA verify cycles might follow a block of DMA read cycles (memory to peripheral) to allow the peripheral to verify its newly acquired data.

### Block Diagram Description

#### 1. DMA Channels

The 8257 provides four separate DMA channels (labeled CH-0 to CH-3). Each channel includes two sixteen-bit registers: (1) a DMA address register, and (2) a terminal count register. Both registers must be initialized before a channel is enabled. The DMA address register is loaded with the address of the first memory location to be accessed. The value loaded into the low-order 14-bits of the terminal count register specifies the number of DMA cycles minus one before the Terminal Count (TC) output is activated. For instance, a terminal count of 0 would cause the TC output to be active in the first DMA cycle for that channel. In general, if N = the number of desired DMA cycles, load the value N-1 into the low-order 14-bits of the terminal count register. The most significant two bits of the terminal count register specify the type of DMA operation for that channel:

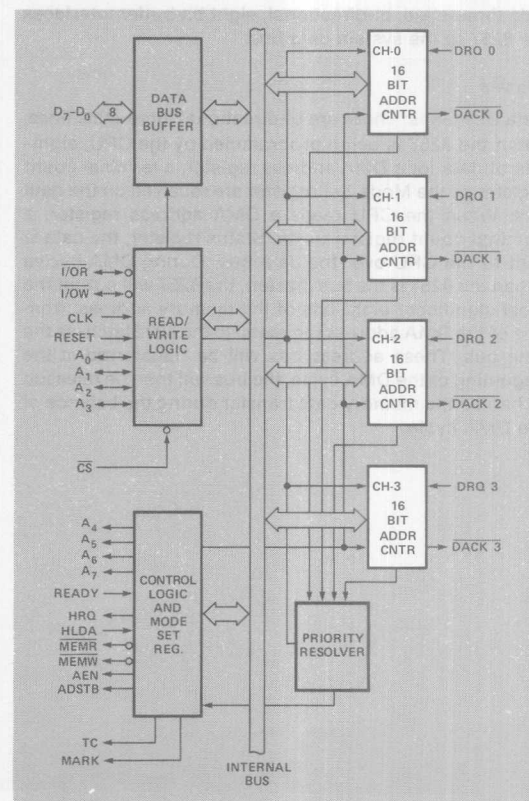


Figure 1. 8257 Block Diagram Showing DMA Channels

These two bits are not modified during a DMA cycle, but can be changed between DMA blocks.

Each channel accepts a DMA Request (DRQn) input and provides a DMA Acknowledge (DACKn) output:

#### (DRQ 0-DRQ 3)

**DMA Request:** These are individual asynchronous channel request inputs used by the peripherals to obtain a DMA cycle. If not in the rotating priority mode then DRQ 0 has the highest priority and DRQ 3 has the lowest. A request can be generated by raising the request line and holding it high until DMA acknowledge. For multiple DMA cycles (Burst Mode) the request line is held high until the DMA acknowledge of the last cycle arrives.

#### (DACK 0 - DACK 3)

**DMA Acknowledge:** An active low level on the acknowledge output informs the peripheral connected to that channel that it has been selected for a DMA cycle.

## 2. Data Bus Buffer

This three-state, bi-directional, eight bit buffer interfaces the 8257 to the system bus:

#### (D<sub>0</sub>-D<sub>7</sub>)

**Data Bus Lines:** These are bi-directional three-state lines. When the 8257 is being programmed by the CPU, eight-bits of data for a DMA address register, a terminal count register or the Mode Set register are received on the data bus. When the CPU reads a DMA address register, a terminal count register or the Status register, the data is sent to the CPU over the data bus. During DMA cycles (when the 8257 is the bus master), the 8257 will output the most significant eight-bits of the memory address (from one of the DMA address registers) to the 8212 latch via the data bus. These address bits will be transferred at the beginning of the DMA cycle; the bus will then be released to handle the memory data transfer during the balance of the DMA cycle.

BIT 15	BIT 14	TYPE OF DMA OPERATION
0	0	Verify DMA Cycle
0	1	Write DMA Cycle
1	0	Read DMA Cycle
1	1	(Illegal)

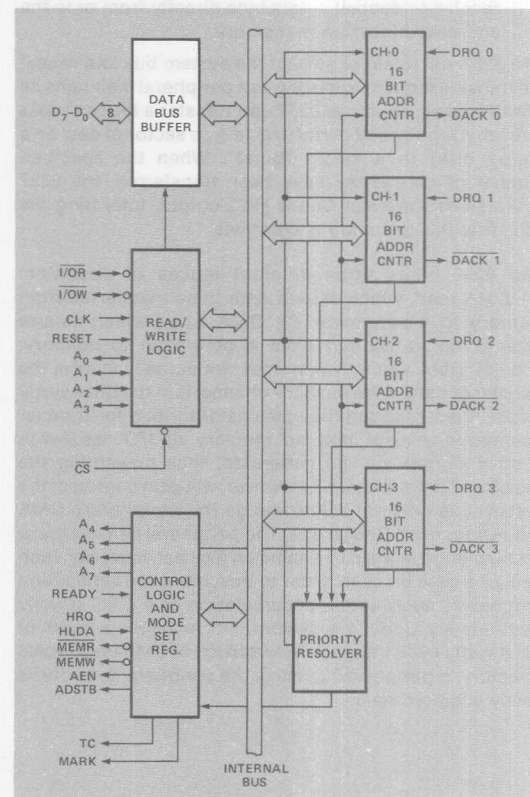


Figure 2. 8257 Block Diagram Showing Data Bus Buffer

### 3. Read/Write Logic

When the CPU is programming or reading one of the 8257's register (i.e., when the 8257 is a "slave" device on the system bus), the Read/Write Logic accepts the I/O Read ( $\overline{I/OR}$ ) or I/O Write ( $\overline{I/OW}$ ) signal, decodes the least significant four address bits, ( $A_0-A_3$ ), and either writes the contents of the data bus into the addressed register (if  $\overline{I/OW}$  is true) or places the contents of the addressed register onto the data bus (if  $\overline{I/OR}$  is true).

During DMA cycles (i.e., when the 8257 is the bus "master"), the Read/Write Logic generates the I/O read and memory write (DMA write cycle) or I/O Write and memory read (DMA read cycle) signals which control the data link with the peripheral that has been granted the DMA cycle.

Note that during DMA transfers Non-DMA I/O devices should be de-selected (disabled) using "AEN" signal to inhibit I/O device decoding of the memory address as an erroneous device address.

#### $\overline{I/OR}$

**I/O Read:** An active-low, bi-directional three-state line. In the "slave" mode, it is an input which allows the 8-bit status register or the upper/lower byte of a 16-bit DMA address register or terminal count register to be read. In the "master" mode,  $\overline{I/OR}$  is a control output which is used to access data from a peripheral during the DMA write cycle.

#### $\overline{I/OW}$

**I/O Write:** An active-low, bi-directional three-state line. In the "slave" mode, it is an input which allows the contents of the data bus to be loaded into the 8-bit mode set register or the upper/lower byte of a 16-bit DMA address register or terminal count register. In the "master" mode,  $\overline{I/OW}$  is a control output which allows data to be output to a peripheral during a DMA read cycle.

#### (CLK)

**Clock Input:** Generally from an Intel® 8224 Clock Generator device. ( $\phi 2$  TTL)

#### (RESET)

**Reset:** An asynchronous input (generally from an 8224 device) which clears all control lines and disables all DMA channels by clearing the mode register.

#### ( $A_0-A_3$ )

**Address Lines:** These least significant four address lines are bi-directional. In the "slave" mode they are inputs which select one of the registers to be read or programmed. In the "master" mode, they are outputs which constitute the least significant four bits of the 16-bit memory address generated by the 8257.

#### ( $\overline{CS}$ )

**Chip Select:** An active-low input which enables the I/O Read or I/O Write input when the 8257 is being read or programmed in the "slave" mode. In the "master" mode,  $\overline{CS}$  is automatically disabled to prevent the chip from selecting itself while performing the DMA function.

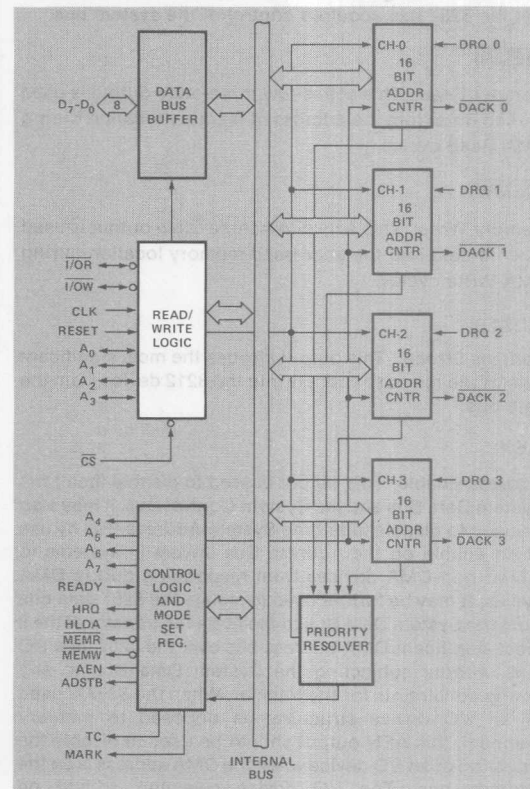


Figure 3. 8257 Block Diagram Showing Read/Write Logic Function

#### 4. Control Logic

This block controls the sequence of operations during all DMA cycles by generating the appropriate control signals and the 16-bit address that specifies the memory location to be accessed.

##### (A<sub>4</sub>-A<sub>7</sub>)

**Address Lines:** These four address lines are three-state outputs which constitute bits 4 through 7 of the 16-bit memory address generated by the 8257 during all DMA cycles.

##### (READY)

**Ready:** This asynchronous input is used to elongate the memory read and write cycles in the 8257 with wait states if the selected memory requires longer cycles.

##### (HRQ)

**Hold Request:** This output requests control of the system bus. In systems with only one 8257, HRQ will normally be applied to the HOLD input on the CPU.

##### (HLDA)

**Hold Acknowledge:** This input from the CPU indicates that the 8257 has acquired control of the system bus.

##### (MEMR)

**Memory Read:** This active-low three-state output is used to read data from the addressed memory location during DMA Read cycles.

##### (MEMW)

**Memory Write:** This active-low three-state output is used to write data into the addressed memory location during DMA Write cycles.

##### (ADSTB)

**Address Strobe:** This output strobes the most significant byte of the memory address into the 8212 device from the data bus.

##### (AEN)

**Address Enable:** This output is used to disable (float) the System Data Bus and the System Control Bus. It may also be used to disable (float) the System Address Bus by use of an enable on the Address Bus drivers in systems to inhibit non-DMA devices from responding during DMA cycles. It may be further used to isolate the 8257 data bus from the System Data Bus to facilitate the transfer of the 8 most significant DMA address bits over the 8257 data I/O pins without subjecting the System Data Bus to any timing constraints for the transfer. When the 8257 is used in an I/O device structure (as opposed to memory mapped), this AEN output should be used to disable the selection of an I/O device when the DMA address is on the address bus. The I/O device selection should be determined by the DMA acknowledge outputs for the 4 channels.

##### (TC)

**Terminal Count:** This output notifies the currently selected peripheral that the present DMA cycle should be the last cycle for this data block. If the TC STOP bit in the Mode Set register is set, the selected channel will be automatically disabled at the end of that DMA cycle. TC is activated when the 14-bit value in the selected channel's terminal count register equals zero. Recall that the low-order 14-bits of the terminal count register should be loaded with the values (n-1), where n = the desired number of the DMA cycles.

##### (MARK)

**Modulo 128 Mark:** This output notifies the selected peripheral that the current DMA cycle is the 128th cycle since the previous MARK output. MARK always occurs at 128 (and all multiples of 128) cycles from the end of the data block. Only if the total number of DMA cycles (n) is evenly divisible by 128 (and the terminal count register was loaded with n-1), will MARK occur at 128 (and each succeeding multiple of 128) cycles from the beginning of the data block.

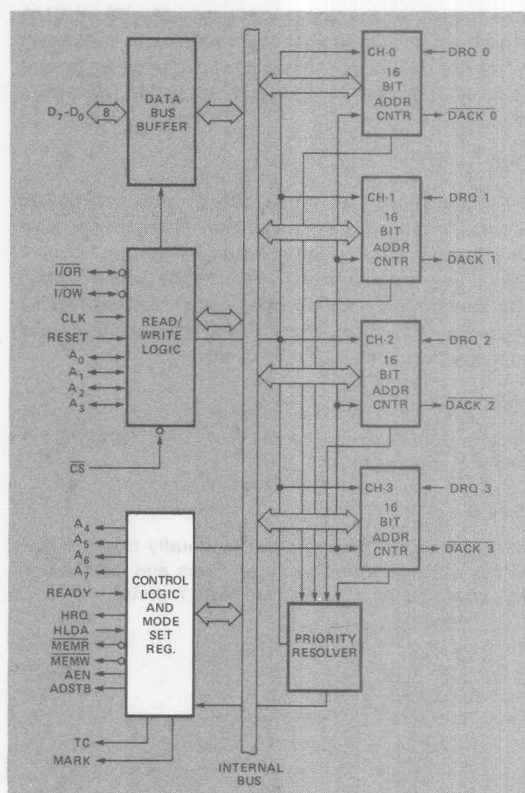
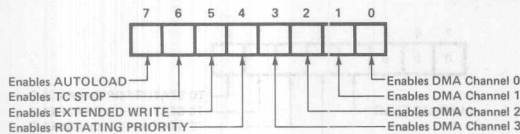


Figure 4. 8257 Block Diagram Showing Control Logic and Mode Set Register

### 5. Mode Set Register

When set, the various bits in the Mode Set register enable each of the four DMA channels, and allow four different options for the 8257:

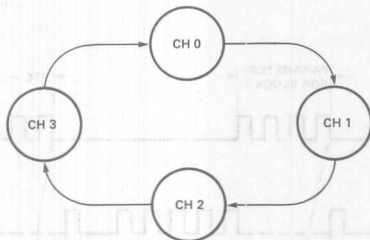


The Mode Set register is normally programmed by the CPU after the DMA address register(s) and terminal count register(s) are initialized. The Mode Set Register is cleared by the RESET input, thus disabling all options, inhibiting all channels, and preventing bus conflicts on power-up. A channel should not be left enabled unless its DMA address and terminal count registers contain valid values; otherwise, an inadvertent DMA request (DRQn) from a peripheral could initiate a DMA cycle that would destroy memory data.

The various options which can be enabled by bits in the Mode Set register are explained below:

#### Rotating Priority Bit 4

In the Rotating Priority Mode, the priority of the channels has a circular sequence. After each DMA cycle, the priority of each channel changes. The channel which has just been serviced will have the lowest priority.



If the ROTATING PRIORITY bit is not set (set to a zero), each DMA channel has a fixed priority. In the fixed priority mode, Channel 0 has the highest priority and Channel 3 has the lowest priority. If the ROTATING PRIORITY bit is set to a one, the priority of each channel changes after each DMA cycle (not each DMA request). Each channel moves up to the next highest priority assignment, while the channel which has just been serviced moves to the lowest priority assignment:

	CHANNEL → JUST SERVICED	CH-0	CH-1	CH-2	CH-3
Priority → Assignments	Highest ↑ Lowest	CH-1 CH-2 CH-3 CH-0	CH-2 CH-3 CH-0 CH-1	CH-3 CH-0 CH-1 CH-2	CH-0 CH-1 CH-2 CH-3

Note that rotating priority will prevent any one channel from monopolizing the DMA mode; consecutive DMA cycles will service different channels if more than one channel is enabled and requesting service. All DMA operations began with Channel 0 initially assigned to the highest priority for the first DMA cycle.

#### Extended Write Bit 5

If the EXTENDED WRITE bit is set, the duration of both the MEMW and I/OW signals is extended by activating them earlier in the DMA cycle. Data transfers within micro-computer systems proceed asynchronously to allow use of various types of memory and I/O devices with different access times. If a device cannot be accessed within a specific amount of time it returns a "not ready" indication to the 8257 that causes the 8257 to insert one or more wait states in its internal sequencing. Some devices are fast enough to be accessed without the use of wait states, but if they generate their READY response with the leading edge of the I/OW or MEMW signal (which generally occurs late in the transfer sequence), they would normally cause the 8257 to enter a wait state because it does not receive READY in time. For systems with these types of devices, the Extended Write option provides alternative timing for the I/O and memory write signals which allows the devices to return an early READY and prevents the unnecessary occurrence of wait states in the 8257, thus increasing system throughput.

#### TC Stop Bit 6

If the TC STOP bit is set, a channel is disabled (i.e., its enable bit is reset) after the Terminal Count (TC) output goes true, thus automatically preventing further DMA operation on that channel. The enable bit for that channel must be re-programmed to continue or begin another DMA operation. If the TC STOP bit is not set, the occurrence of the TC output has no effect on the channel enable bits. In this case, it is generally the responsibility of the peripheral to cease DMA requests in order to terminate a DMA operation.

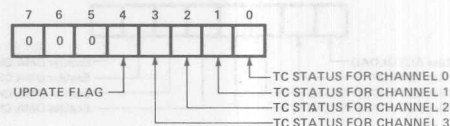
#### Auto Load Bit 7

The Auto Load mode permits Channel 2 to be used for repeat block or block chaining operations, without immediate software intervention between blocks. Channel 2 registers are initialized as usual for the first data block; Channel 3 registers, however, are used to store the block re-initialization parameters (DMA starting address, terminal count and DMA transfer mode). After the first block of DMA cycles is executed by Channel 2 (i.e., after the TC output goes true), the parameters stored in the Channel 3 registers are transferred to Channel 2 during an "update" cycle. Note that the TC STOP feature, described above, has no effect on Channel 2 when the Auto Load bit is set.

Channel 2 are automatically duplicated in the Channel 3 registers when Channel 2 is programmed. This permits repeat block operations to be set up with the programming of a single channel. Repeat block operations can be used in applications such as CRT refreshing. Channels 2 and 3 can still be loaded with separate values if Channel 2 is loaded before loading Channel 3. Note that in the Auto Load mode, Channel 3 is still available to the user if the Channel 3 enable bit is set, but use of this channel will change the values to be auto loaded into Channel 2 at update time. All that is necessary to use the Auto Load feature for chaining operations is to reload Channel 3 registers at the conclusion of each update cycle with the new parameters for the next data block transfer.

Each time that the 8257 enters an update cycle, the update flag in the status register is set and parameters in Channel 3 are transferred to Channel 2, non-destructively for Channel 3. The actual re-initialization of Channel 2 occurs at the beginning of the next channel 2 DMA cycle after the TC cycle. This will be the first DMA cycle of the new data block for Channel 2. The update flag is cleared at the conclusion of this DMA cycle. For chaining operations, the update flag in the status register can be monitored by the CPU to determine when the re-initialization process has been completed so that the next block parameters can be safely loaded into Channel 3.

The eight-bit status register indicates which channels have reached a terminal count condition and includes the update flag described previously.



The TC status bits are set when the Terminal Count (TC) output is activated for that channel. These bits remain set until the status register is read or the 8257 is reset. The UPDATE FLAG, however, is not affected by a status register read operation. The UPDATE FLAG can be cleared by resetting the 8257, by changing to the non-auto load mode (i.e., by resetting the AUTO LOAD bit in the Mode Set register) or it can be left to clear itself at the completion of the update cycle. The purpose of the UPDATE FLAG is to prevent the CPU from inadvertently skipping a data block by overwriting a starting address or terminal count in the Channel 3 registers before those parameters are properly auto-loaded into Channel 2.

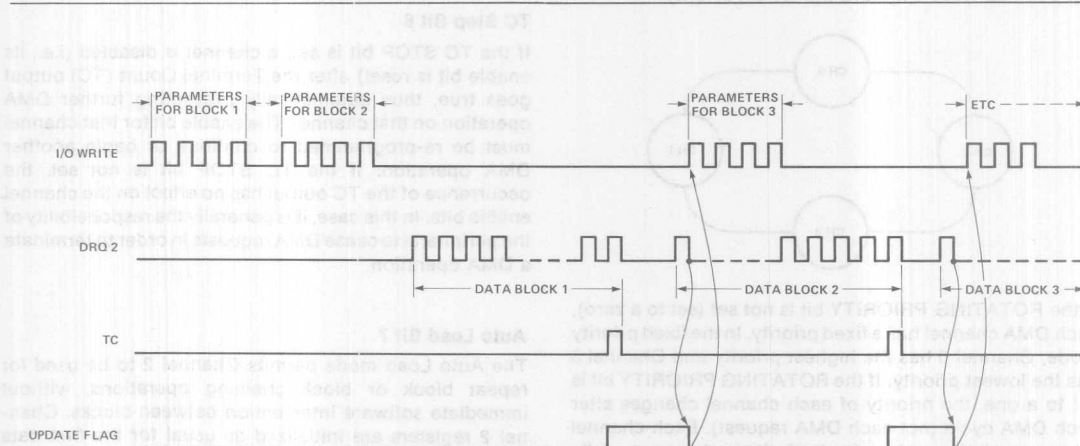


Figure 5. Autoload Timing

## OPERATIONAL SUMMARY

### Programming and Reading the 8257 Registers

There are four pairs of "channel registers": each pair consisting of a 16-bit DMA address register and a 16-bit terminal count register (one pair for each channel). The 8257 also includes two "general registers": one 8-bit Mode Set register and one 8-bit Status register. The registers are loaded or read when the CPU executes a write or read instruction that addresses the 8257 device and the appropriate register within the 8257. The 8228 generates the appropriate read or write control signal (generally I/OR or I/OW while the CPU places a 16-bit address on the system address bus, and either outputs the data to be written onto the system data bus or accepts the data being read from the data bus. All or some of the most significant 12 address bits A<sub>4</sub>-A<sub>15</sub> (depending on the systems memory, I/O configuration) are usually decoded to produce the chip select (CS) input to the 8257. An I/O Write input (or Memory Write in memory mapped I/O configurations, described below) specifies that the addressed register is to be programmed, while an I/O Read input (or Memory Read) specifies that the addressed register is to be read. Address bit 3 specifies whether a "channel register" (A<sub>3</sub> = 0) or the Mode Set (program only)/Status (read only) register (A<sub>3</sub> = 1) is to be accessed.

The least significant three address bits, A<sub>0</sub>-A<sub>2</sub>, indicate the specific register to be accessed. When accessing the Mode Set or Status register, A<sub>0</sub>-A<sub>2</sub> are all zero. When accessing a channel register bit A<sub>0</sub> differentiates between the DMA address register (A<sub>0</sub> = 0) and the terminal count register (A<sub>0</sub> = 1), while bits A<sub>1</sub> and A<sub>2</sub> specify one of the

CONTROL INPUT	$\overline{CS}$	I/OW	I/OR	A <sub>3</sub>
Program Half of a Channel Register	0	0	1	0
Read Half of a Channel Register	0	1	0	0
Program Mode Set Register	0	0	1	1
Read Status Register	0	1	0	1

four channels. Because the "channel registers" are 16-bits, two program instruction cycles are required to load or read an entire register. The 8257 contains a first/last (F/L) flip flop which toggles at the completion of each channel program or read operation. The F/L flip flop determines whether the upper or lower byte of the register is to be accessed. The F/L flip flop is reset by the RESET input and whenever the Mode Set register is loaded. To maintain proper synchronization when accessing the "channel registers" all channel command instruction operations should occur in pairs, with the lower byte of a register always being accessed first. Do not allow  $\overline{CS}$  to clock while either I/OR or I/OW is active, as this will cause an erroneous F/L flip flop state. In systems utilizing an interrupt structure, interrupts should be disabled prior to any paired programming operations to prevent an interrupt from splitting them. The result of such a split would leave the F/L F/F in the wrong state. This problem is particularly obvious when other DMA channels are programmed by an interrupt structure.

### 8257 Register Selection

REGISTER	BYTE	ADDRESS INPUTS				F/L	*BI-DIRECTIONAL DATA BUS							
		A <sub>3</sub>	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>		D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CH-0 DMA Address	LSB	0	0	0	0	0	A <sub>7</sub>	A <sub>6</sub>	A <sub>5</sub>	A <sub>4</sub>	A <sub>3</sub>	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>
	MSB	0	0	0	0	1	A <sub>15</sub>	A <sub>14</sub>	A <sub>13</sub>	A <sub>12</sub>	A <sub>11</sub>	A <sub>10</sub>	A <sub>9</sub>	A <sub>8</sub>
CH-0 Terminal Count	LSB	0	0	0	1	0	C <sub>7</sub>	C <sub>6</sub>	C <sub>5</sub>	C <sub>4</sub>	C <sub>3</sub>	C <sub>2</sub>	C <sub>1</sub>	C <sub>0</sub>
	MSB	0	0	0	1	1	Rd	Wr	C <sub>13</sub>	C <sub>12</sub>	C <sub>11</sub>	C <sub>10</sub>	C <sub>9</sub>	C <sub>8</sub>
CH-1 DMA Address	LSB	0	0	1	0	0	Same as Channel 0							
	MSB	0	0	1	0	1								
CH-1 Terminal Count	LSB	0	0	1	1	0	Same as Channel 0							
	MSB	0	0	1	1	1								
CH-2 DMA Address	LSB	0	1	0	0	0	Same as Channel 0							
	MSB	0	1	0	0	1								
CH-2 Terminal Count	LSB	0	1	0	1	0	Same as Channel 0							
	MSB	0	1	0	1	1								
CH-3 DMA Address	LSB	0	1	1	0	0	Same as Channel 0							
	MSB	0	1	1	0	1								
CH-3 Terminal Count	LSB	0	1	1	1	0	Same as Channel 0							
	MSB	0	1	1	1	1								
MODE SET (Program only)	—	1	0	0	0	0	AL	TCS	EW	RP	EN3	EN2	EN1	EN0
STATUS (Read only)	—	1	0	0	0	0	0	0	0	UP	TC3	TC2	TC1	TC0

\*A<sub>0</sub>-A<sub>15</sub>: DMA Starting Address, C<sub>0</sub>-C<sub>13</sub>: Terminal Count value (N-1), Rd and Wr: DMA Verify (00), Write (01) or Read (10) cycle selection, AL: Auto Load, TCS: TC STOP, EW: EXTENDED WRITE, RP: ROTATING PRIORITY, EN3-EN0: CHANNEL ENABLE MASK, UP: UPDATE FLAG, TC3-TC0: TERMINAL COUNT STATUS BITS.

### DMA Operation

Internal 8257 operations may proceed through seven different states. The duration of a state is defined by the clock input. When the 8257 is not executing a DMA cycle, it is in the idle state,  $S_1$ . A DMA cycle begins when one or more DMA Request ( $DRQ_n$ ) lines become active. The 8257 then enters state  $S_0$ , sends a Hold Request ( $HRQ$ ) to the CPU and waits for as many  $S_0$  states as are necessary for the CPU to return a Hold Acknowledge ( $HLDA$ ). For each  $S_0$  state, the DMA Request lines are again sampled and DMA priority is resolved (according to the fixed or rotating priority scheme). When  $HLDA$  is received, the DMA Acknowledge ( $DACK_n$ ) line for the highest priority requesting channel is activated, thus selecting that channel and its peripheral for the DMA cycle. The 8257 then proceeds to state  $S_1$ . Note that the DMA Request ( $DRQ_n$ ) input should remain high until either  $DACK_n$  is received for a single DMA cycle service, or until both the  $DACK_n$  and  $TC$  outputs are received when transferring an entire data block in a "burst" mode. If the 8257 should lose control of the system bus (i.e., if  $HLDA$  goes false), the DMA Acknowledge will be removed after the current DMA cycle is completed and no more DMA cycles will occur until the 8257 again acquires control of the system bus.

Each DMA cycle will consist of at least four internal states:  $S_1$ ,  $S_2$ ,  $S_3$ , and  $S_4$ . If the access time for the memory or I/O devices involved is not fast enough to return the required  $READY$  response and complete a byte transfer within the specified amount of time, one or more wait states ( $SW$ ) are inserted between states  $S_3$  and  $S_4$ . Recall that in certain cases the Extended Write option can eliminate the need for a wait state. Note that a  $READY$  response is not required during DMA verify cycles. Specified minimum/maximum values for  $READY$  setup time ( $t_{RS}$ ), write data setup time ( $t_{DW}$ ), read data access time ( $t_{RD}$ ) and  $HLDA$  setup time ( $t_{QS}$ ) are listed under A.C. CHARACTERISTICS and are illustrated in the accompanying timing diagrams.

During DMA write cycles, the I/O Read ( $I/OR$ ) output is generated at the beginning of state  $S_2$  and the Memory Write ( $MEMW$ ) output is generated at the beginning of  $S_3$ . During DMA read cycles, the Memory Read ( $MEMR$ ) output is generated at the beginning of state  $S_2$  and the I/O Write ( $I/OW$ ) output goes true at the beginning of state  $S_3$ . Recall that no read or write control signals are generated during DMA verify cycles. Extended  $WR$  for MEM and I/O will be generated in  $S_2$ .

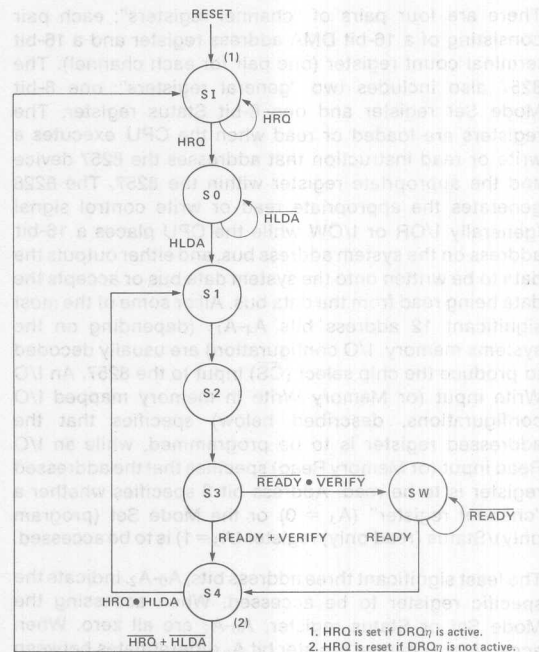


Figure 6. DMA Operation State Diagram

REGISTER		BYTE		ADDRESS INPUTS	
				A <sub>1</sub>	A <sub>0</sub>
CH0 DMA Address	LSB	0	0	0	0
	MID	0	0	0	0
	MSB	0	0	0	0
CH0 Terminal Count	LSB	0	0	0	0
	MID	0	0	0	0
	MSB	0	0	0	0
CH1 DMA Address	LSB	0	0	0	1
	MID	0	0	0	1
	MSB	0	0	0	1
CH1 Terminal Count	LSB	0	0	0	1
	MID	0	0	0	1
	MSB	0	0	0	1
CH2 DMA Address	LSB	0	0	0	1
	MID	0	0	0	1
	MSB	0	0	0	1
CH2 Terminal Count	LSB	0	0	0	1
	MID	0	0	0	1
	MSB	0	0	0	1
CH3 DMA Address	LSB	0	0	0	1
	MID	0	0	0	1
	MSB	0	0	0	1
CH3 Terminal Count	LSB	0	0	0	1
	MID	0	0	0	1
	MSB	0	0	0	1
MODE SET (Program only)		--	--	1	0
STATUS (Read only)		--	--	1	0

### Memory Mapped I/O Configurations

The 8257 can be connected to the system bus as a memory device instead of as an I/O device for memory mapped I/O configurations by connecting the system memory control lines to the 8257's I/O control lines and the system I/O control lines to the 8257's memory control lines.

This configuration permits use of the 8080's considerably larger repertoire of memory instructions when reading or loading the 8257's registers. Note that with this connection, the programming of the Read (bit 15) and Write (bit 14) bits in the terminal count register will have a different meaning:

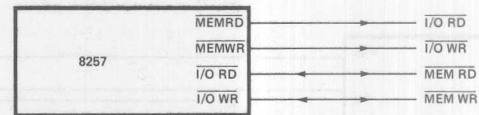


Figure 7. System Interface for Memory Mapped I/O

BIT 15 READ	BIT 14 WRITE	
0	0	DMA Verify Cycle
0	1	DMA Read Cycle
1	0	DMA Write Cycle
1	1	Illegal

Figure 8. TC Register for Memory Mapped I/O Only

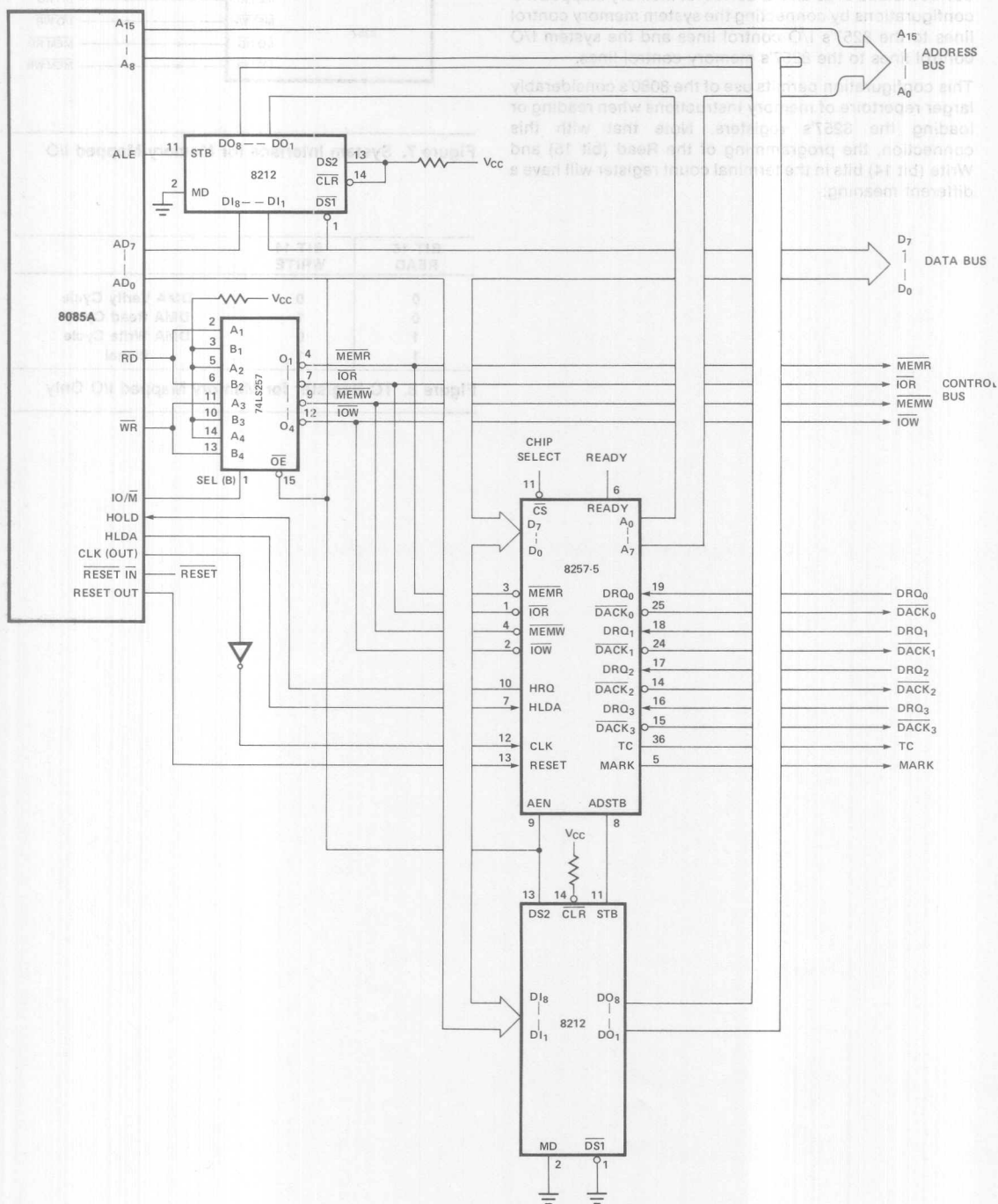


Figure 9. Detailed System Interface Schematic

**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature Under Bias. . . . . 0°C to 70°C  
 Storage Temperature . . . . . -65°C to +150°C  
 Voltage on Any Pin  
   With Respect to Ground . . . . . -0.5V to +7V  
 Power Dissipation . . . . . 1 Watt

*\*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

**D.C. CHARACTERISTICS**

$T_A = 0^\circ\text{C to } 70^\circ\text{C}$ ,  $V_{CC} = +5\text{V} \pm 5\%$ ,  $\text{GND} = 0\text{V}$

SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
$V_{IL}$	Input Low Voltage	-0.5	0.8	Volts	
$V_{IH}$	Input High Voltage	2.0	$V_{CC} + 5$	Volts	
$V_{OL}$	Output Low Voltage		0.45	Volts	$I_{OL} = 1.6 \text{ mA}$
$V_{OH}$	Output High Voltage	2.4	$V_{CC}$	Volts	$I_{OH} = -150 \mu\text{A}$ for AB, DB and AEN $I_{OH} = -80 \mu\text{A}$ for others
$V_{HH}$	HRQ Output High Voltage	3.3	$V_{CC}$	Volts	$I_{OH} = -80 \mu\text{A}$
$I_{CC}$	$V_{CC}$ Current Drain		120	mA	
$I_{IL}$	Input Leakage		$\pm 10$	$\mu\text{A}$	$V_{IN} = V_{CC}$ to 0V
$I_{OFL}$	Output Leakage During Float		$\pm 10$	$\mu\text{A}$	$V_{OUT} = V_{CC}$ to 0V

**CAPACITANCE**

$T_A = 25^\circ\text{C}$ ;  $V_{CC} = \text{GND} = 0\text{V}$

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNIT	TEST CONDITIONS
$C_{IN}$	Input Capacitance			10	pF	$f_c = 1 \text{ MHz}$
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured pins returned to GND

**PRELIMINARY**  
 Notice: This is not a final specification. Some  
 parametric limits are subject to change.

## A.C. CHARACTERISTICS: PRIPHERAL (SLAVE) MODE

$T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{CC} = 5.0\text{V} \pm 5\%$ ;  $GND = 0\text{V}$  (Note 1).

### 8080 Bus Parameters

#### Read Cycle:

Symbol	Parameter	8257		8257-5		Unit	Test Conditions
		Min.	Max.	Min.	Max.		
$T_{AR}$	Adr or $\overline{CS}$ Setup to $\overline{RD}$ ↓	0		0		ns	
$T_{RA}$	Adr or $\overline{CS}$ ↑ Hold from $\overline{RD}$ ↑	0		0		ns	
$T_{RD}$	Data Access from $\overline{RD}$ ↓	0	300	0	200	ns	(Note 2)
$T_{DF}$	DB→Float Delay from $\overline{RD}$ ↑	20	150	20	100	ns	
$T_{RR}$	$\overline{RD}$ Width	250		250		ns	

#### Write Cycle:

Symbol	Parameter	8257		8257-5		Unit	Test Conditions
		Min.	Max.	Min.	Max.		
$T_{AW}$	Adr Setup to $\overline{WR}$ ↓	20		20		ns	
$T_{WA}$	Adr Hold from $\overline{WR}$ ↑	0		0		ns	
$T_{DW}$	Data Setup to $\overline{WR}$ ↑	200		200		ns	
$T_{WD}$	Data Hold from $\overline{WR}$ ↑	0		0		ns	
$T_{WW}$	$\overline{WR}$ Width	200		200		ns	

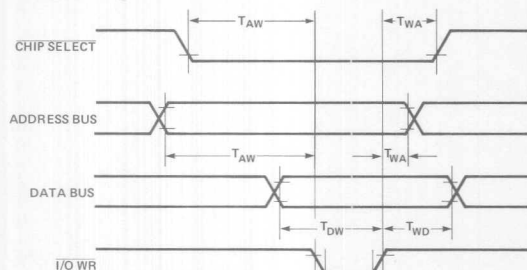
#### Other Timing:

Symbol	Parameter	8257		8257-5		Unit	Test Conditions
		Min.	Max.	Min.	Max.		
$T_{RSTW}$	Reset Pulse Width	300		300		ns	
$T_{RSTD}$	Power Supply↑ ( $V_{CC}$ ) Setup to Reset↓	500		500		μs	
$T_r$	Signal Rise Time		20		20	ns	
$T_f$	Signal Fall Time		20		20	ns	
$T_{RSTS}$	Reset to First $\overline{IOWR}$	2		2		tcy	

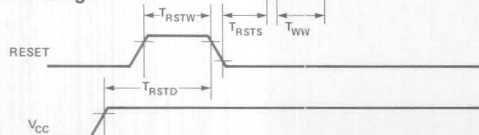
Notes: 1. All timing measurements are made at the following reference voltages unless specified otherwise: Input "1" at 2.0V, "0" at 0.8V  
 2. 8257:  $C_L = 100\text{pF}$ , 8257-5:  $C_L = 150\text{pF}$ .  
 Output "1" at 2.0V, "0" at 0.8V

## 8257 PERIPHERAL MODE TIMING DIAGRAMS

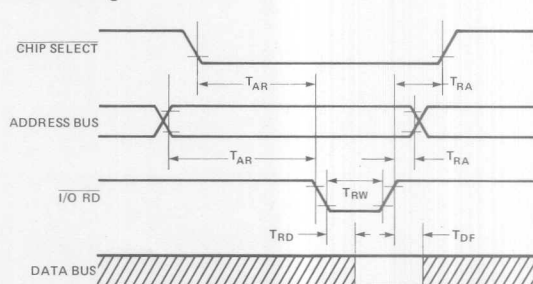
#### Write Timing:



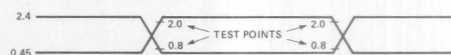
#### Reset Timing:



#### Read Timing:



#### Input Waveform for A.C. Tests:

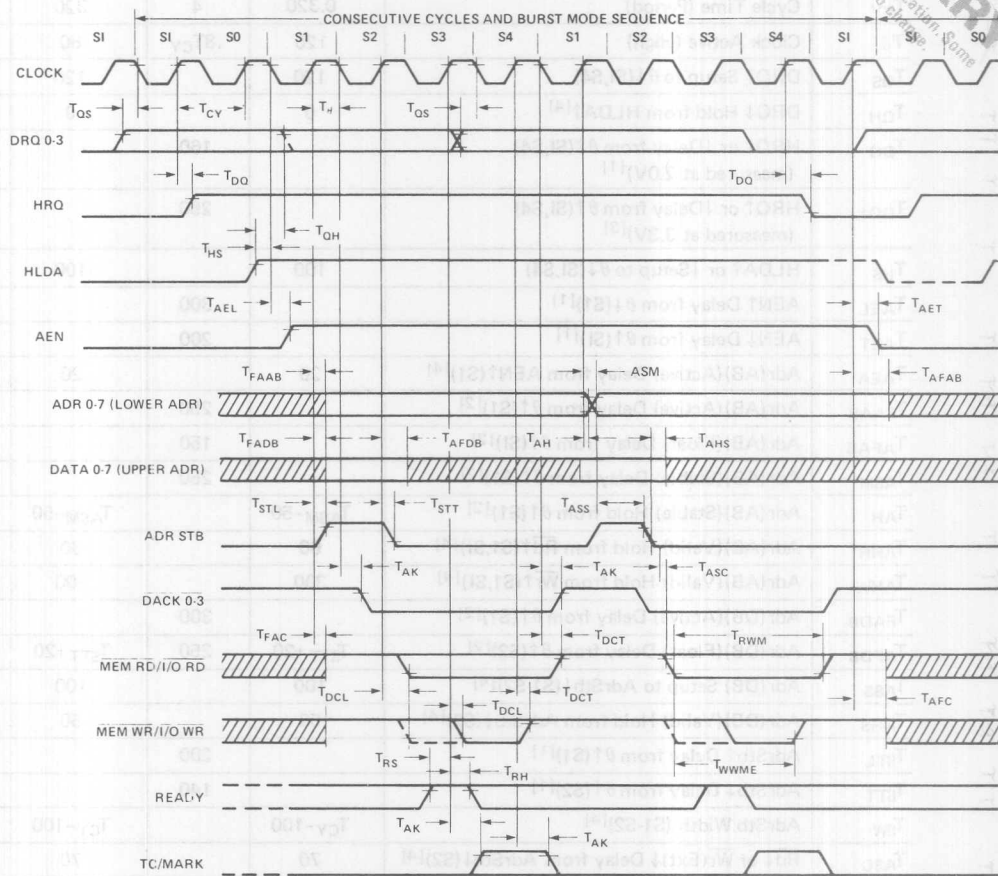


A.C. CHARACTERISTICS: DMA (MASTER) MODE  $T_A = 0^\circ\text{C to } 70^\circ\text{C}$ ,  $V_{CC} = +5V \pm 5\%$ ,  $GND = 0V$ 

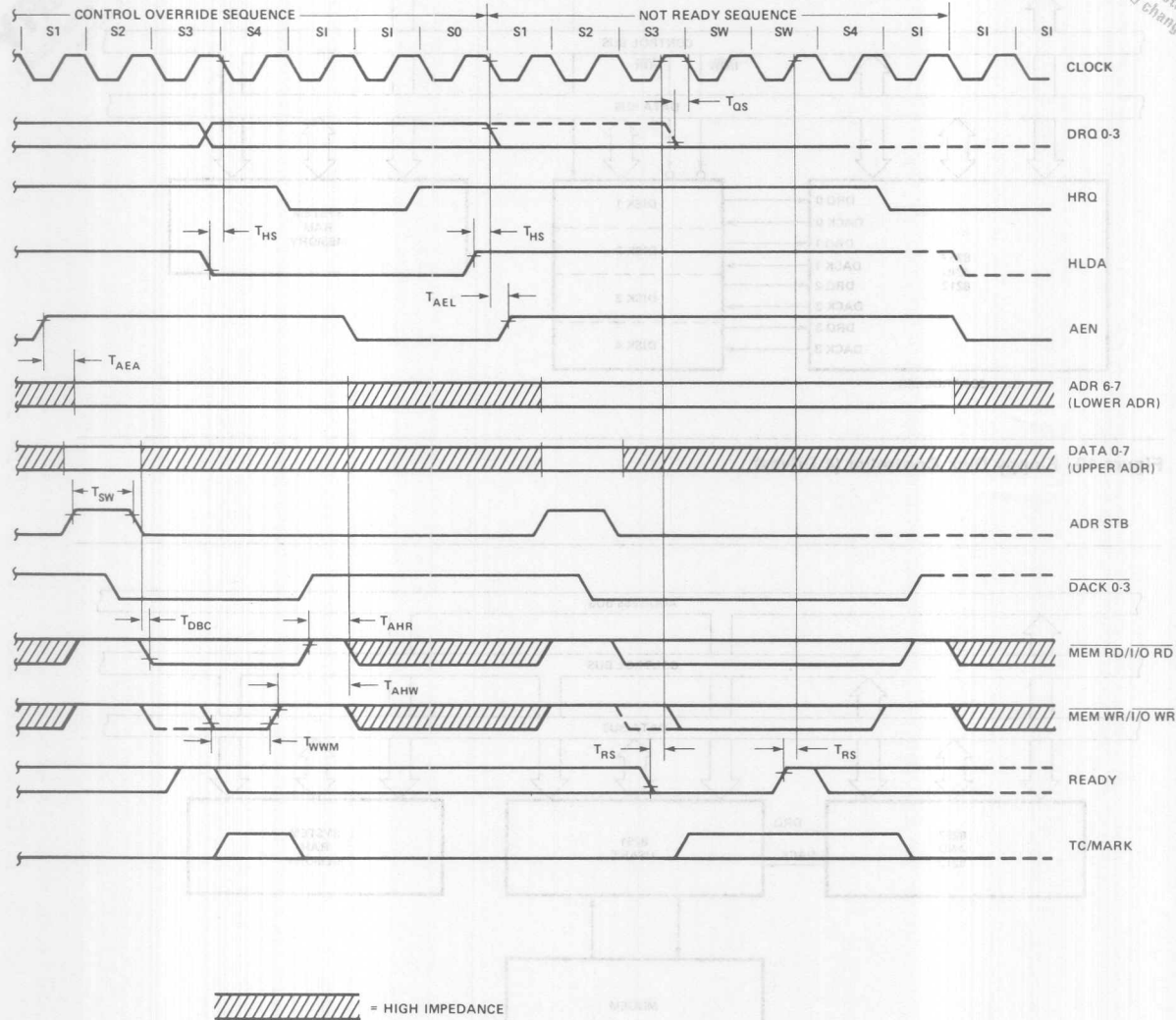
SYMBOL	PARAMETER	8257		8257-5		UNIT
		MIN.	MAX.	MIN.	MAX.	
$T_{CY}$	Cycle Time (Period)	0.320	4	320	4	$\mu\text{s}$
$T_\theta$	Clock Active (High)	120	.8 $T_{CY}$	80	.8 $T_{CY}$	ns
$T_{QS}$	DRQ $\uparrow$ Setup to $\theta\downarrow$ (S1,S4)	120		120		
$T_{QH}$	DRQ $\downarrow$ Hold from HLDA $\uparrow$ [4]	0		0		
$T_{DQ}$	HRQ $\uparrow$ or $\downarrow$ Delay from $\theta\uparrow$ (S1,S4) (measured at 2.0V)[1]		160		160	ns
$T_{DQ1}$	HRQ $\uparrow$ or $\downarrow$ Delay from $\theta\uparrow$ (S1,S4) (measured at 3.3V)[3]		250		250	ns
$T_{HS}$	HLDA $\uparrow$ or $\downarrow$ Setup to $\theta\downarrow$ (S1,S4)	100		100		ns
$T_{AEL}$	AEN $\uparrow$ Delay from $\theta\downarrow$ (S1)[1]		300		300	ns
$T_{AET}$	AEN $\downarrow$ Delay from $\theta\uparrow$ (S1)[1]		200		200	ns
$T_{AEA}$	Adr(AB)(Active) Delay from AEN $\uparrow$ (S1)[4]	20		20		ns
$T_{FAAB}$	Adr(AB)(Active) Delay from $\theta\uparrow$ (S1)[2]		250		250	ns
$T_{AFAB}$	Adr(AB)(Float) Delay from $\theta\uparrow$ (S1)[2]		150		150	ns
$T_{ASM}$	Adr(AB)(Stable) Delay from $\theta\uparrow$ (S1)[2]		250		250	ns
$T_{AH}$	Adr(AB)(Stable) Hold from $\theta\uparrow$ (S1)[2]	$T_{ASM}-50$		$T_{ASM}-50$		
$T_{AHR}$	Adr(AB)(Valid) Hold from $\bar{Rd}\uparrow$ (S1,S1)[4]	60		60		ns
$T_{AHW}$	Adr(AB)(Valid) Hold from $\bar{Wr}\uparrow$ (S1,S1)[4]	300		300		ns
$T_{FADB}$	Adr(DB)(Active) Delay from $\theta\uparrow$ (S1)[2]		300		300	ns
$T_{AFDB}$	Adr(DB)(Float) Delay from $\theta\uparrow$ (S2)[2]	$T_{STT}+20$	250	$T_{STT}+20$	170	ns
$T_{ASS}$	Adr(DB) Setup to AdrStb $\downarrow$ (S1-S2)[4]	100		100		ns
$T_{AHS}$	Adr(DB)(Valid) Hold from AdrStb $\downarrow$ (S2)[4]	50		50		ns
$T_{STL}$	AdrStb $\uparrow$ Delay from $\theta\uparrow$ (S1)[1]		200		200	ns
$T_{STT}$	AdrStb $\downarrow$ Delay from $\theta\uparrow$ (S2)[1]		140		140	ns
$T_{SW}$	AdrStb Width (S1-S2)[4]	$T_{CY}-100$		$T_{CY}-100$		ns
$T_{ASC}$	$\bar{Rd}\downarrow$ or $\bar{Wr}$ (Ext) $\downarrow$ Delay from AdrStb $\downarrow$ (S2)[4]	70		70		ns
$T_{DBC}$	$\bar{Rd}\downarrow$ or $\bar{Wr}$ (Ext) $\downarrow$ Delay from Adr(DB) (Float)(S2)[4]	20		20		ns
$T_{AK}$	DAK $\uparrow$ or $\downarrow$ Delay from $\theta\downarrow$ (S2,S1) and TC/Mark $\uparrow$ Delay from $\theta\uparrow$ (S3) and TC/Mark $\downarrow$ Delay from $\theta\uparrow$ (S4)[1,5]		250		250	ns
$T_{DCL}$	$\bar{Rd}\downarrow$ or $\bar{Wr}$ (Ext) $\downarrow$ Delay from $\theta\uparrow$ (S2) and $\bar{Wr}\downarrow$ Delay from $\theta\uparrow$ (S3)[2,6]		200		200	ns
$T_{DCT}$	$\bar{Rd}\uparrow$ Delay from $\theta\downarrow$ (S1,S1) and $\bar{Wr}\uparrow$ Delay from $\theta\uparrow$ (S4)[2,7]		200		200	ns
$T_{FAC}$	$\bar{Rd}$ or $\bar{Wr}$ (Active) from $\theta\uparrow$ (S1)[2]		300		300	ns
$T_{AFC}$	$\bar{Rd}$ or $\bar{Wr}$ (Float) from $\theta\uparrow$ (S1)[2]		150		150	ns
$T_{RWM}$	$\bar{Rd}$ Width (S2-S1 or S1)[4]	$2T_{CY} + T_\theta - 50$		$2T_{CY} + T_\theta - 50$		ns
$T_{WWW}$	$\bar{Wr}$ Width (S3-S4)[4]	$T_{CY}-50$		$T_{CY}-50$		ns
$T_{WWME}$	$\bar{Wr}$ (Ext) Width (S2-S4)[4]	$2T_{CY}-50$		$2T_{CY}-50$		ns
$T_{RS}$	READY Set Up Time to $\theta\uparrow$ (S3, Sw)	30		30		ns
$T_{RH}$	READY Hold Time from $\theta\uparrow$ (S3, Sw)	20		20		ns

Notes: 1. Load = 1 TTL. 2. Load = 1 TTL + 50pF. 3. Load = 1 TTL + ( $R_L = 3.3K$ ),  $V_{OH} = 3.3V$ . 4. Tracking Specification.  
5.  $\Delta T_{AK} < 50$  ns. 6.  $\Delta T_{DCL} < 50$  ns. 7.  $\Delta T_{DCT} < 50$  ns.

## DMA MODE WAVEFORMS



**PRELIMINARY**  
 Notice: This is not a final specification. Some  
 parametric limits are subject to change.



**PRELIMINARY**  
 Notice: This is not a final specification. Some  
 parametric limits are subject to change.

## SYSTEM APPLICATION EXAMPLES

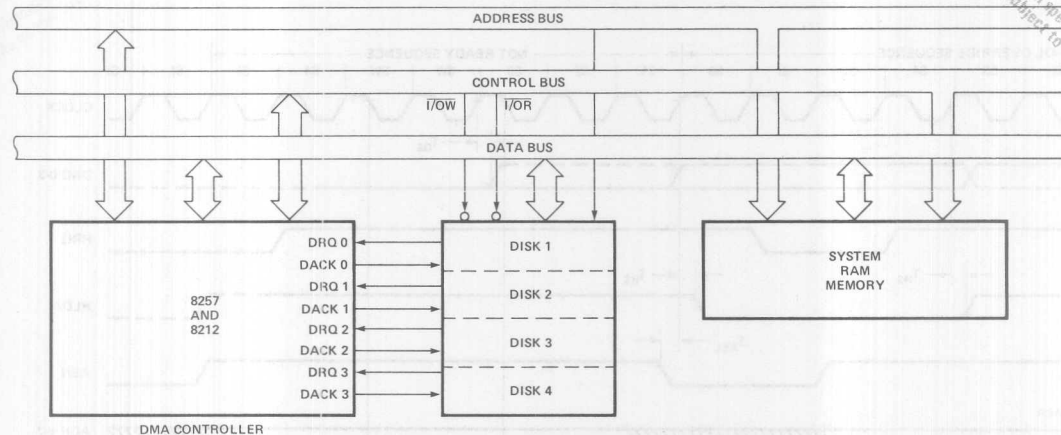


Figure 10. Floppy Disk Controller (4 Drives)

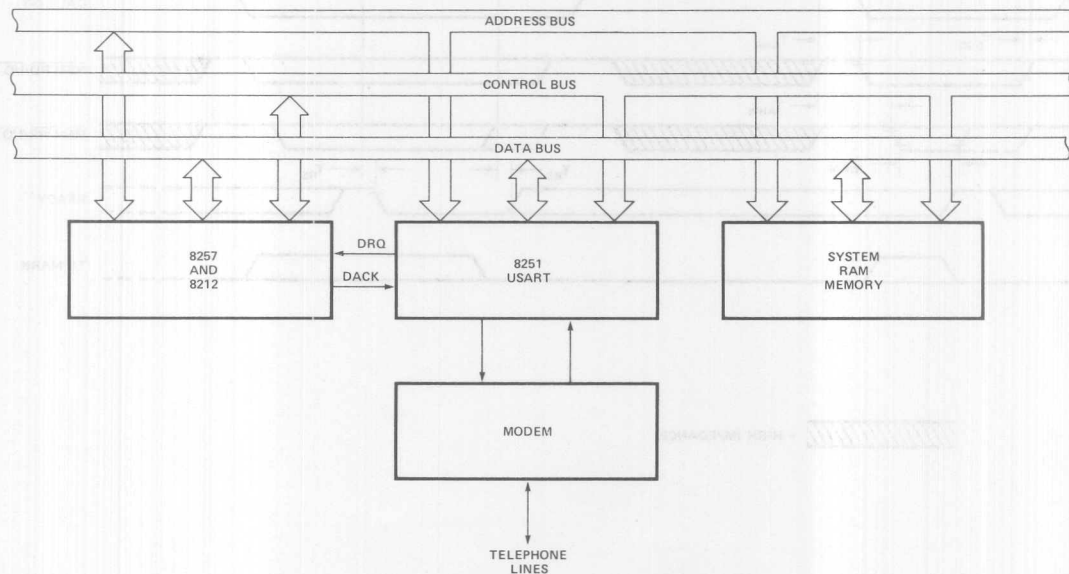


Figure 11. High-Speed Communication Controller

# 8259/8259-5

## PROGRAMMABLE INTERRUPT CONTROLLER

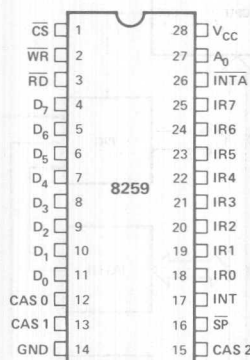
**PRELIMINARY**  
Notice: This is not a final specification. Some parametric limits are subject to change.

- MCS-85™ Compatible 8259-5
- 8-Level Priority Controller
- Expandable to 64 Levels
- Programmable Interrupt Modes
- Individual Request Mask Capability
- Single +5V Supply (No Clocks)
- 28-Pin Dual In-Line Package

The Intel® 8259 handles up to 8 vectored priority interrupts for the CPU. It is cascadable for up to 64 vectored priority interrupts, without additional circuitry. It will be packaged in a 28-pin plastic DIP, uses nMOS technology and requires a single +5V supply. Circuitry is static, requiring no clock input.

The 8259 is designed to minimize the software and real time overhead in handling multi-level priority interrupts. It has several modes, permitting optimization for a variety of system requirements.

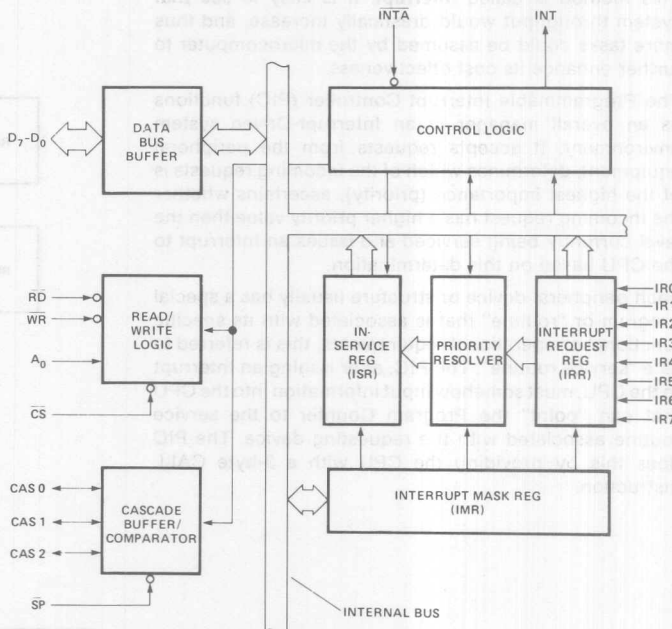
### PIN CONFIGURATION



### PIN NAMES

D <sub>7</sub> -D <sub>0</sub>	DATA BUS (BI-DIRECTIONAL)
RD	READ INPUT
WR	WRITE INPUT
A <sub>0</sub>	COMMAND SELECT ADDRESS
CS	CHIP SELECT
CAS1-CAS0	CASCADE LINES
SP	SLAVE PROGRAM INPUT
INT	INTERRUPT OUTPUT
INTA	INTERRUPT ACKNOWLEDGE INPUT
IR0-IR7	INTERRUPT REQUEST INPUTS

### BLOCK DIAGRAM



## INTRODUCTION TO THE USE OF INTERRUPTS IN MICROCOMPUTER SYSTEMS

Microcomputer system design requires that I/O devices such as keyboards, displays, sensors and other components receive servicing in an efficient method so that large amounts of the total system tasks can be assumed by the microcomputer with little or no effect on throughput.

The most common method of servicing such devices is the **Polled** approach. This is where the processor must test each device in sequence and in effect "ask" each one if it needs servicing. It is easy to see that a large portion of the main program is looping through this continuence polling cycle and that such a method would have a serious, detrimental effect on system throughput thus limiting the tasks that could be assumed by the microcomputer and reducing the cost effectiveness of using such devices.

A more desirable method would be one that would allow the microprocessor to be executing its main program and only stop to service peripheral devices when it is told to do so by the device itself. In effect, the method would provide an external asynchronous input that would inform the processor that it should complete whatever instruction that is currently being executed and fetch a new routine that will service the requesting device. Once this servicing is complete however the processor would resume exactly where it left off.

This method is called **Interrupt**. It is easy to see that system throughput would drastically increase, and thus more tasks could be assumed by the microcomputer to further enhance its cost effectiveness.

The Programmable Interrupt Controller (PIC) functions as an overall manager in an Interrupt-Driven system environment. It accepts requests from the peripheral equipment, determines which of the incoming requests is of the highest importance (priority), ascertains whether the incoming request has a higher priority value than the level currently being serviced and issues an Interrupt to the CPU based on this determination.

Each peripheral device or structure usually has a special program or "routine" that is associated with its specific functional or operational requirements; this is referred to as a "service routine". The PIC, after issuing an Interrupt to the CPU, must somehow input information into the CPU that can "point" the Program Counter to the service routine associated with the requesting device. The PIC does this by providing the CPU with a 3-byte CALL instruction.

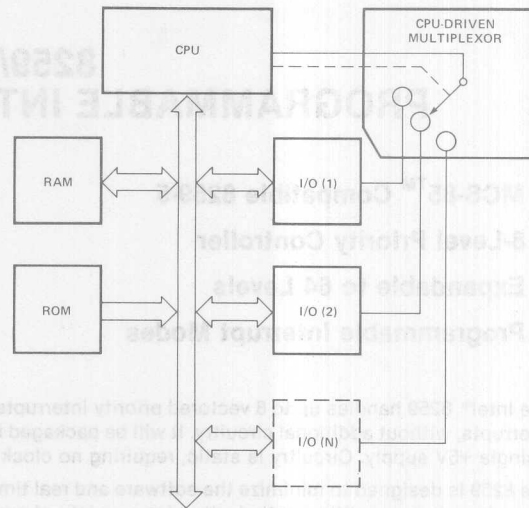


Figure 1. Polled Method

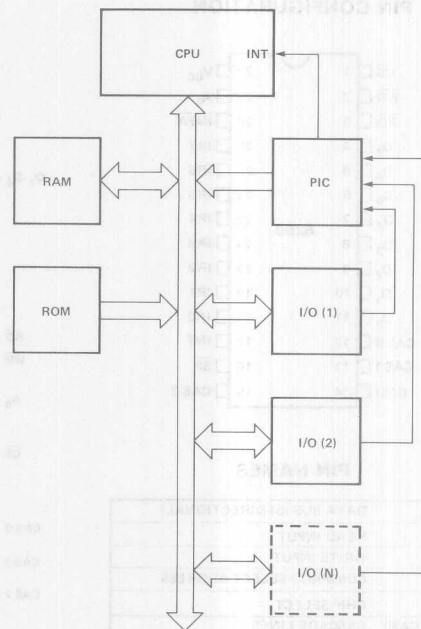


Figure 2. Interrupt Method

## FUNCTIONAL DESCRIPTION

### General

The 8259 is a device specifically designed for use in real time, interrupt driven, microcomputer systems. It manages eight levels or requests and has built-in features for expandability to other 8259s (up to 64 levels). It is programmed by the system's software as an I/O peripheral. A selection of priority modes is available to the programmer so that the manner in which the requests are processed by the 8259 can be configured to match his system requirements. The priority modes can be changed or reconfigured dynamically at any time during the main program. This means that the complete interrupt structure can be defined as required, based on the total system environment.

### Interrupt Request Register (IRR) and In-Service Register (ISR)

The interrupts at the IR input lines are handled by two registers in cascade, the Interrupt Request Register (IRR) and the In-Service Register (ISR). The IRR is used to store all the interrupt levels which are requesting service; and the ISR is used to store all the interrupt levels which are being serviced.

### Priority Resolver

This logic block determines the priorities of the bits set in the IRR. The highest priority is selected and strobed into the corresponding bit of the ISR during INTA pulse.

### INT (Interrupt)

This output goes directly to the CPU interrupt input. The  $V_{OH}$  level on this line is designed to be fully compatible with the 8080 input level.

### INTA (Interrupt Acknowledge)

Three INTA pulses will cause the 8259 to release a 3-byte CALL instruction onto the Data Bus.

### Interrupt Mask Register (IMR)

The IMR stores the bits of the interrupt lines to be masked. The IMR operates on the ISR. Masking of a higher priority input will not affect the interrupt request lines of lower priority.

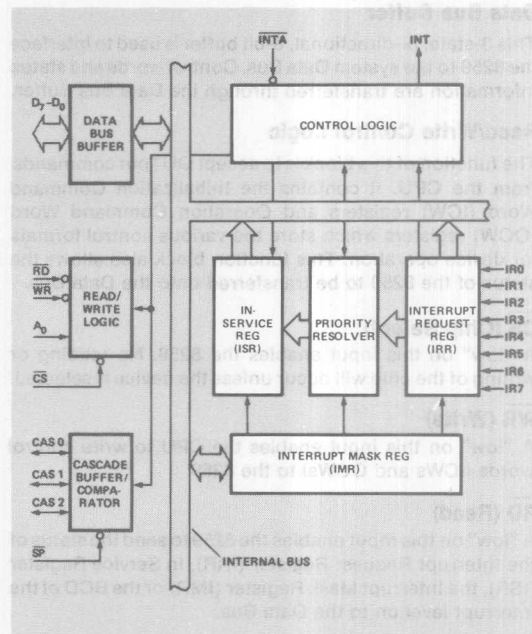


Figure 5. 8259 Block Diagram Showing Basic Interrupt Functions

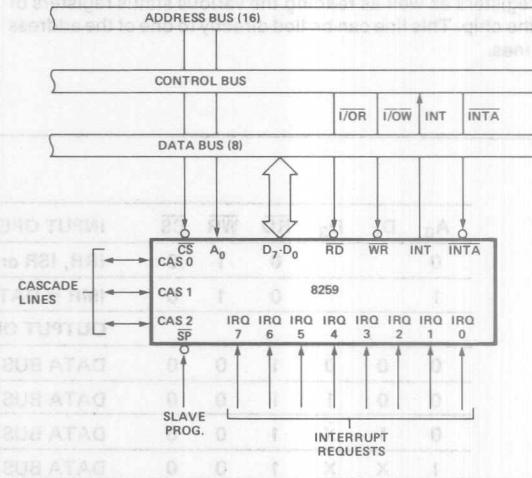


Figure 6. 8259 Interface to Standard System Bus

### Data Bus Buffer

This 3-state, bi-directional, 8-bit buffer is used to interface the 8259 to the system Data Bus. Control words and status information are transferred through the Data Bus Buffer.

### Read/Write Control Logic

The function of this block is to accept OUTput commands from the CPU. It contains the Initialization Command Word (ICW) registers and Operation Command Word (OCW) registers which store the various control formats for device operation. This function block also allows the status of the 8259 to be transferred onto the Data Bus.

### CS (Chip Select)

A "low" on this input enables the 8259. No reading or writing of the chip will occur unless the device is selected.

### WR (Write)

A "low" on this input enables the CPU to write control words (ICWs and OCWs) to the 8259.

### RD (Read)

A "low" on this input enables the 8259 to send the status of the Interrupt Request Register (IRR), In Service Register (ISR), the Interrupt Mask Register (IMR) or the BCD of the Interrupt level on to the Data Bus.

### A<sub>0</sub>

This input signal is used in conjunction with WR and RD signals to write commands into the various command registers as well as reading the various status registers of the chip. This line can be tied directly to one of the address lines.

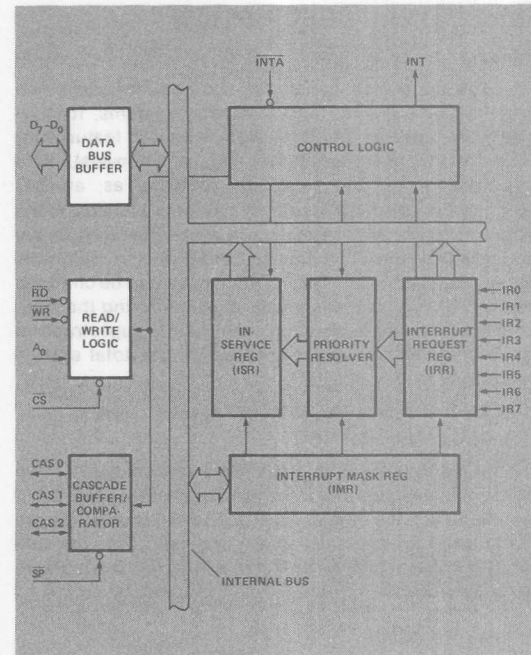


Figure 3. 8259 Block Diagram Showing Data Bus Buffer and Read/Write Logic Functions

A <sub>0</sub>	D <sub>4</sub>	D <sub>3</sub>	RD	WR	CS	INPUT OPERATION (READ)
0			0	1	0	IRR, ISR or Interrupting Level ⇒ DATA BUS (Note 1)
1			0	1	0	IMR ⇒ DATA BUS
OUTPUT OPERATION (WRITE)						
0	0	0	1	0	0	DATA BUS ⇒ OCW2
0	0	1	1	0	0	DATA BUS ⇒ OCW3
0	1	X	1	0	0	DATA BUS ⇒ ICW1
1	X	X	1	0	0	DATA BUS ⇒ OCW1, ICW2, ICW3 (Note 2)
DISABLE FUNCTION						
X	X	X	1	1	0	DATA BUS ⇒ 3-STATE
X	X	X	X	X	1	DATA BUS ⇒ 3-STATE

Note 1: Selection of IRR, ISR or Interrupting Level is based on the content of OCW3 written before the READ operation.

Note 2: On-chip sequencer logic queues these commands into proper sequence.

Figure 4. 8259 Basic Operation

### SP (Slave Program)

More than one 8259 can be used in the system to expand the priority interrupt scheme up to 64 levels. In such case, one 8259 acts as the master, and the others act as slaves. A "high" on the  $\overline{SP}$  pin designates the 8259 as the master, a "low" designates it as a slave.

### The Cascade/Buffer/Comparator

This function block stores and compares the IDs of all 8259 used in the system. The associated three I/O pins (CAS0-2) are outputs when the 8259 is used as a master ( $\overline{SP} = 1$ ), and are inputs when the 8259 is used as a slave ( $\overline{SP} = 0$ ). As a master, the 8259 sends the ID of the interrupting slave device onto the CAS0-2 lines. The slave thus selected will send its preprogrammed subroutine address onto the Data Bus during next two consecutive  $\overline{INTA}$  pulses. (See section "Cascading the 8259".)

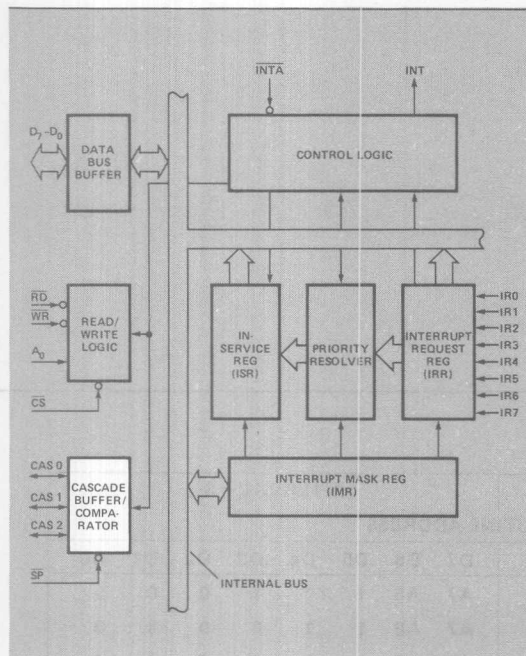


Figure 7. 8259 Block Diagram Showing Cascading Function

## OPERATIONAL DESCRIPTION

### General

The powerful features of the 8259 in a microcomputer system are its programmability and its utilization of the CALL instruction to jump into any address in the memory map. The normal sequence of events that the 8259 interacts with the CPU is as follows:

1. One or more of the INTERRUPT REQUEST lines (IR7-0) are raised high, setting the corresponding IRR bit(s).
2. The 8259 accepts these requests, resolves the priorities, and sends an INT to the CPU.

3. The CPU acknowledges the INT and responds with an  $\overline{INTA}$  pulse.
4. Upon receiving an  $\overline{INTA}$  from the CPU group, the highest priority ISR bit is set, and the corresponding IRR bit is reset. The 8259 will also release a CALL instruction code (11001101) onto the 8-bit Data Bus through its D7-0 pins.
5. This CALL instruction will initiate two more  $\overline{INTA}$  pulses to be sent to the 8259 from the CPU group.
6. These two  $\overline{INTA}$  pulses allow the 8259 to release its preprogrammed subroutine address onto the Data Bus. The lower 8-bit address is released at the first  $\overline{INTA}$  pulse and the higher 8-bit address is released at the second  $\overline{INTA}$  pulse.
7. This completes the 3-byte CALL instruction released by the 8259. ISR bit is not reset until the end of the subroutine when an EOI (End of interrupt) command is issued to the 8259.

### Programming The 8259

The 8259 accepts two types of command words generated by the CPU:

1. Initialization Command Words (ICWs):

Before normal operation can begin, each 8259 in the system must be brought to a starting point — by a sequence of 2 or 3 bytes timed by  $\overline{WR}$  pulses. This sequence is described in Figure 1.

2. Operation Command Words (OCWs):

These are the command words which command the 8259 to operate in various interrupt modes. These modes are:

- a. Fully nested mode
- b. Rotating priority mode
- c. Special mask mode
- d. Polled mode

The OCWs can be written into the 8259 at anytime after initialization.

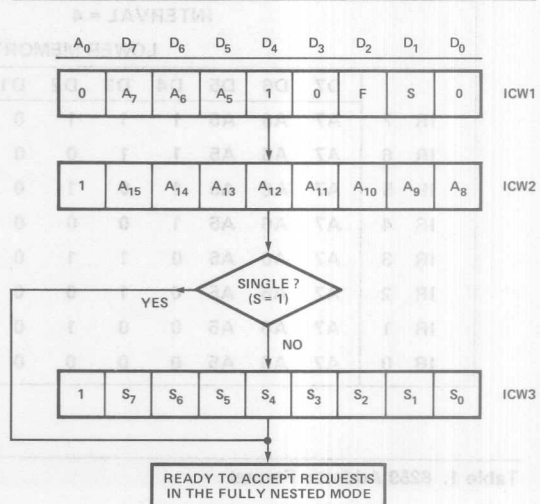


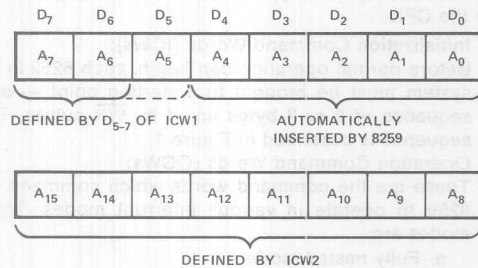
Figure 8. Initialization Sequence

Whenever a command is issued with A0 = 0 and D4 = 1, this is interpreted as Initialization Command Word 1 (ICW1), and initiates the initialization sequence. During this sequence, the following occur automatically:

- The edge sense circuit is reset, which means that following initialization, an interrupt request (IR) input must make a low to high transition to generate an interrupt.
- The Interrupt Mask Register is cleared.
- IR 7 input is assigned priority 7.
- Special Mask Mode Flip-flop and status Read Flip-flop are reset.

The 8 requesting devices have 8 addresses equally spaced in memory. The addresses can be programmed at intervals of 4 or 8 bytes; the 8 routines thus occupying a page of 32 or 64 bytes respectively in memory.

The address format is:



ICW1. Thus, the interrupt service routines can be located anywhere in the memory space. The 8 byte interval will maintain compatibility with current 8080 RESTART instruction software, while the 4 byte interval is best for compact jump table.

The address format inserted by the 8259 is described in Table 1.

The bits F and S are defined by ICW1 as follows:

F: Call address interval. F = 1, then interval = 4; F = 0, then interval = 8.

S: Single. S = 1 means that this is the only 8259 in the system. It avoids the necessity of programming ICW3.

	INTERVAL = 4								INTERVAL = 8							
	LOWER MEMORY ROUTINE ADDRESS															
	D7	D6	D5	D4	D3	D2	D1	D0	D7	D6	D5	D4	D3	D2	D1	D0
IR 7	A7	A6	A5	1	1	1	0	0	A7	A6	1	1	1	0	0	0
IR 6	A7	A6	A5	1	1	0	0	0	A7	A6	1	1	0	0	0	0
IR 5	A7	A6	A5	1	0	1	0	0	A7	A6	1	0	1	0	0	0
IR 4	A7	A6	A5	1	0	0	0	0	A7	A6	1	0	0	0	0	0
IR 3	A7	A6	A5	0	1	1	0	0	A7	A6	0	1	1	0	0	0
IR 2	A7	A6	A5	0	1	0	0	0	A7	A6	0	1	0	0	0	0
IR 1	A7	A6	A5	0	0	1	0	0	A7	A6	0	0	1	0	0	0
IR 0	A7	A6	A5	0	0	0	0	0	A7	A6	0	0	0	0	0	0

Table 1. 8259 Address Format

**Example of Interrupt Acknowledge Sequence**

Assume the 8259 is programmed with F = 1 (CALL address interval = 4), and IR5 is the interrupting level. The 3 byte sequence released by the 8259 timed by the INTA pulses is as follows:

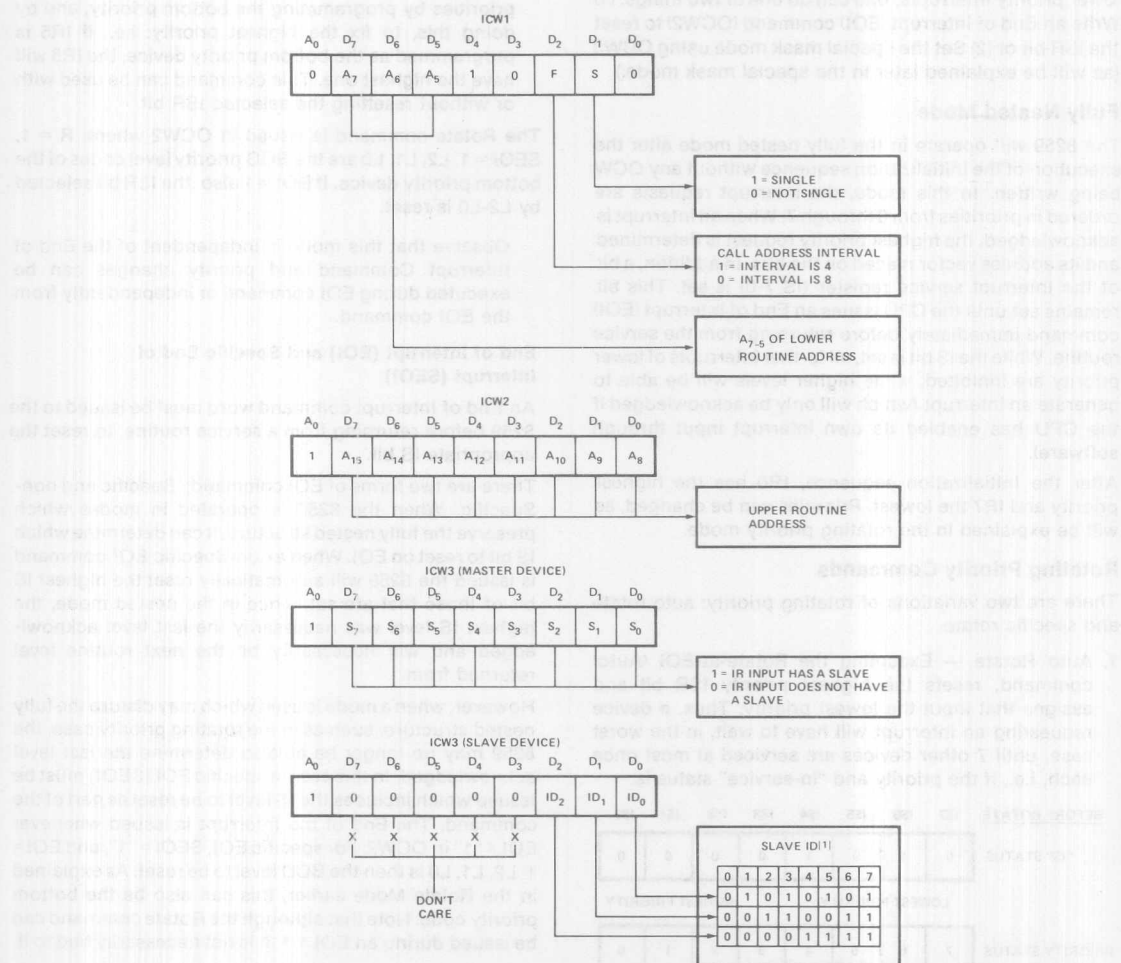
	D7	D6	D5	D4	D3	D2	D1	D0	
1st INTA	1	1	0	0	1	1	0	1	CALL CODE
2nd INTA	A7	A6	A5	1	0	1	0	0	LOWER ROUTINE ADDRESS
3rd INTA	A15	A14	A13	A12	A11	A10	A9	A8	HIGHER ROUTINE ADDRESS

**Initialization Command Word 3 (ICW3)**

This will load the 8-bit slave register. The functions of this register are as follows:

- If the 8259 is the master, a "1" is set for each slave in the system. The master then will release byte 1 of the CALL sequence and will enable the corresponding slave to release bytes 2 and 3, through the cascade lines.
- If the 8259 is a slave, bits 2 - 0 identify the slave. The slave compares its CASO-2 inputs (sent by the master) with these bits. If they are equal, bytes 2 and 3 of the CALL sequence are released.

If bit S is set in ICW1, there is no need to program ICW3.



NOTE 1: SLAVE ID IS EQUAL TO THE CORRESPONDING MASTER IR INPUT.

**Figure 9. Initialization Command Word Format**

## Operation Command Words (OCWs)

After the Initialization Command Words (ICWs) are programmed into the 8259, the chip is ready to accept interrupt requests at its input lines. However, during the 8259 operation, a selection of algorithms can command the 8259 to operate in various modes through the Operation Command Words (OCWs). These various modes and their associated OCWs are described below.

### Interrupt Masks

Each Interrupt Request input can be masked individually by the Interrupt Masked Register (IMR) programmed through OCW1.

The IMR operates on the In-Service Register. Note that if an interrupt is already acknowledged by the 8259 (an INTA pulse has occurred), then the Interrupting level, although masked, will inhibit the lower priorities. To enable these lower priority interrupts, one can do one of two things: (1) Write an End of Interrupt (EOI) command (OCW2) to reset the ISR bit or (2) Set the special mask mode using OCW3 (as will be explained later in the special mask mode.)

### Fully Nested Mode

The 8259 will operate in the fully nested mode after the execution of the initialization sequence without any OCW being written. In this mode, the interrupt requests are ordered in priorities from 0 through 7. When an interrupt is acknowledged, the highest priority request is determined and its address vector placed on the bus. In addition, a bit of the Interrupt service register (IS 7-0) is set. This bit remains set until the CPU issues an End of Interrupt (EOI) command immediately before returning from the service routine. While the IS bit is set, all further interrupts of lower priority are inhibited, while higher levels will be able to generate an interrupt (which will only be acknowledged if the CPU has enabled its own interrupt input through software).

After the Initialization sequence, IR0 has the highest priority and IR7 the lowest. Priorities can be changed, as will be explained in the rotating priority mode.

### Rotating Priority Commands

There are two variations of rotating priority: auto rotate and specific rotate.

1. Auto Rotate — Executing the Rotate-at-EOI (Auto) command, resets the highest priority ISR bit and assigns that input the lowest priority. Thus, a device requesting an interrupt will have to wait, in the worst case, until 7 other devices are serviced at most once each, i.e., if the priority and "in-service" status is:

BEFORE ROTATE	IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
"IS" STATUS	0	1	0	1	0	0	0	0
	LOWEST PRIORITY				HIGHEST PRIORITY			
PRIORITY STATUS	7	6	5	4	3	2	1	0

AFTER ROTATE	IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
"IS" STATUS	0	1	0	0	0	0	0	0
	LOWEST PRIORITY				HIGHEST PRIORITY			
PRIORITY STATUS	4	3	2	1	0	7	6	5

In this example, the In-Service FF corresponding to line 4 (the highest priority FF set) was reset and line 4 became the lowest priority, while all the other priorities rotated correspondingly.

The Rotate command is issued in OCW2, where: R = 1, EOI = 1, SEOI = 0.

2. Specific Rotate — The programmer can change priorities by programming the bottom priority, and by doing this, to fix the highest priority: i.e., if IR5 is programmed as the bottom priority device, the IR6 will have the highest one. This command can be used with or without resetting the selected ISR bit.

The Rotate command is issued in OCW2 where: R = 1, SEOI = 1. L2, L1, L0 are the BCD priority level codes of the bottom priority device. If EOI = 1 also, the ISR bit selected by L2-L0 is reset.

Observe that this mode is independent of the End of Interrupt Command and priority changes can be executed during EOI command or independently from the EOI command.

### End of Interrupt (EOI) and Specific End of Interrupt (SEOI)

An End of Interrupt command word must be issued to the 8259 before returning from a service routine, to reset the appropriate IS bit.

There are two forms of EOI command: Specific and non-Specific. When the 8259 is operated in modes which preserve the fully nested structure, it can determine which IS bit to reset on EOI. When a non-Specific EOI command is issued the 8259 will automatically reset the highest IS bit of those that are set, since in the nested mode, the highest IS level was necessarily the last level acknowledged and will necessarily be the next routine level returned from.

However, when a mode is used which may disturb the fully nested structure, such as in the rotating priority case, the 8259 may no longer be able to determine the last level acknowledged. In this case, a specific EOI (SEOI) must be issued which includes the IS level to be reset as part of the command. The End of the Interrupt is issued whenever EOI = "1" in OCW2. For specific EOI, SEOI = "1", and EOI = 1. L2, L1, L0 is then the BCD level to be reset. As explained in the Rotate Mode earlier, this can also be the bottom priority code. Note that although the Rotate command can be issued during an EOI = 1, it is not necessarily tied to it.

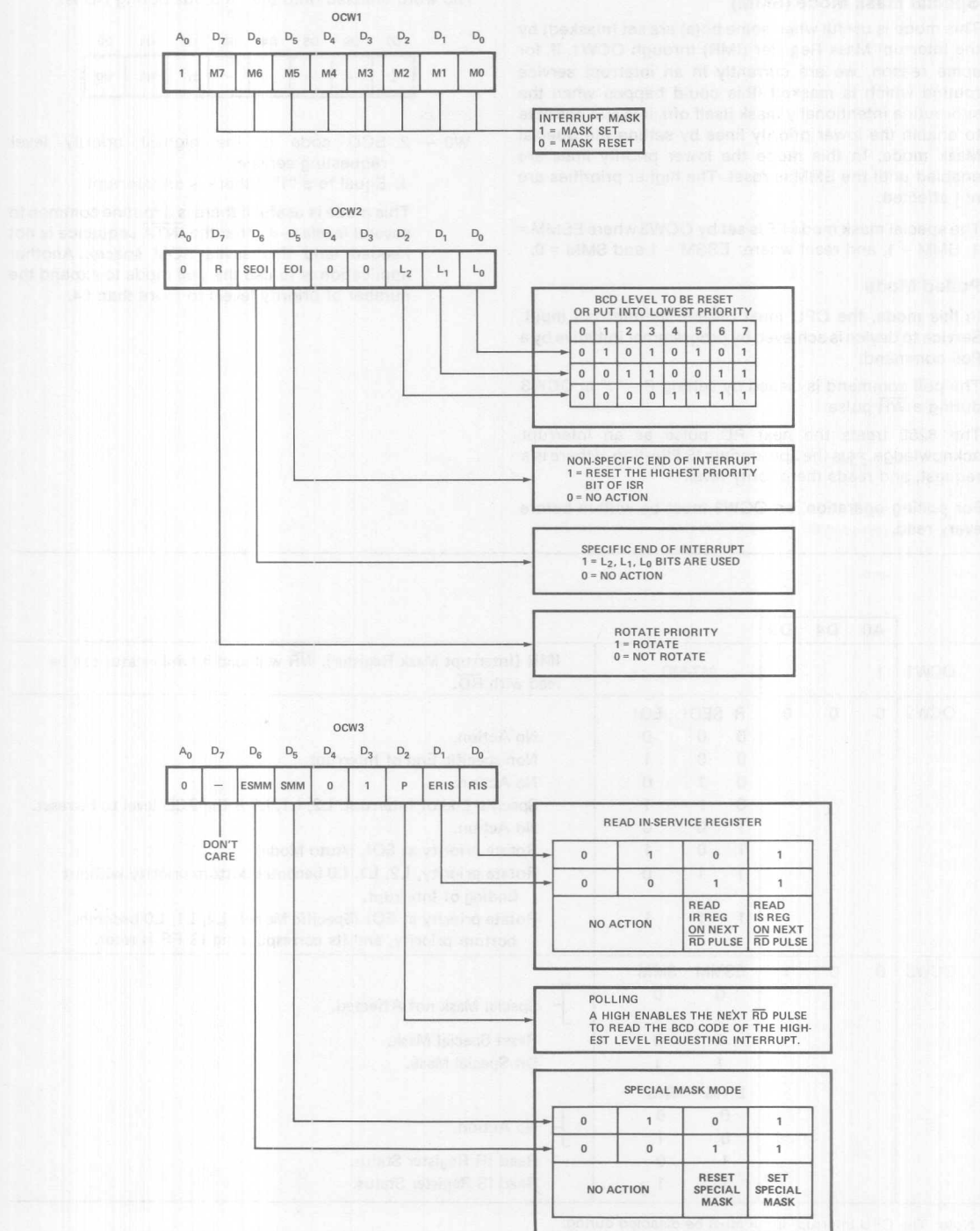


Figure 10. Operation Command Word Format

### Special Mask Mode (SMM)

This mode is useful when some bit(s) are set (masked) by the Interrupt Mask Register (IMR) through OCW1. If, for some reason, we are currently in an interrupt service routine which is masked (this could happen when the subroutine intentionally mask itself off), it is still possible to enable the lower priority lines by setting the Special Mask mode. In this mode the lower priority lines are enabled until the SMM is reset. The higher priorities are not affected.

The special mask mode FF is set by OCW3 where ESMM = 1, SMM = 1, and reset where: ESSM = 1 and SMM = 0.

### Polled Mode

In this mode, the CPU must disable its interrupt input. Service to device is achieved by programmer initiative by a Poll command.

The poll command is issued by setting P = "1" in OCW3 during a  $\overline{WR}$  pulse.

The 8259 treats the next  $\overline{RD}$  pulse as an interrupt acknowledge, sets the appropriate IS Flip-flop, if there is a request, and reads the priority level.

For polling operation, an OCW3 must be written before every read.

The word enabled onto the data bus during  $\overline{RD}$  is:

D7	D6	D5	D4	D3	D2	D1	D0
I	—	—	—	—	W2	W1	W0

W0 — 2: BCD code of the highest priority level requesting service.

I: Equal to a "1" if there is an interrupt.

This mode is useful if there is a routine common to several levels — so that the  $\overline{INTA}$  sequence is not needed (and this saves ROM space). Another application is to use the poll mode to expand the number of priority levels to more than 64.

	A0	D4	D3		
OCW1	1			M7-M0	IMR (Interrupt Mask Register). $\overline{WR}$ will load it while status can be read with $\overline{RD}$ .
OCW2	0	0	0	R SEOI EOI 0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 1 0 1 1 1 0 1 1 1	No Action. Non-specific End of Interrupt. No Action. Specific End of Interrupt. L2, L1, L0 is the BCD level to be reset. No Action. Rotate priority at EOI. (Auto Mode) Rotate priority, L2, L1, L0 becomes bottom priority without Ending of Interrupt. Rotate priority at EOI (Specific Mode), L2, L1, L0 becomes bottom priority, and its corresponding IS FF is reset.
OCW3	0	0	1	ESMM SMM 0 0 0 1 1 0 1 1 ERIS RIS 0 0 0 1 1 0 1 1	Special Mask not Affected. Reset Special Mask. Set Special Mask. No Action. Read IR Register Status. Read IS Register Status.

Note: The CPU interrupt input must be disabled during:

1. Initialization sequence for all the 8259 in the system.
2. Any control command execution.

Figure 11. Summary of Operation Word Programming

### Reading 8259 Status

The input status of several internal registers can be read to update the user information on the system. The following registers can be read by issuing a suitable OCW3 and reading with  $\overline{RD}$ .

**Interrupt Requests Register (IRR):** 8-bit register which contains the levels requesting an interrupt to be acknowledged. The highest request level is reset from the IRR when an interrupt is acknowledged. (Not affected by IMR).

**In Service Register (ISR):** 8-bit register which contains the priority levels that are being serviced. The ISR is updated when an End of Interrupt command is issued.

**Interrupt Mask Register:** 8-bit register which contains the interrupt request lines which are masked.

The IRR can be read when prior to the  $\overline{RD}$  pulse, an  $\overline{WR}$  pulse is issued with OCW3, and ERIS = 1, RIS = 0.

The ISR can be read in a similar mode, when ERIS = 1, RIS = 1.

There is no need to write an OCW3 before every status read operation as long as the status read corresponds with the previous one, i.e. the 8259 "remembers" whether the IRR or ISR has been previously selected by the OCW3.

For reading the IMR, a  $\overline{WR}$  pulse is not necessary to precede the  $\overline{RD}$ . The output data bus will contain the IMR whenever  $\overline{RD}$  is active and A0 = 1.

Polling overrides status read when P = 1, ERIS = 1 in OCW3.

### Cascading

The 8259 can be easily interconnected in a system of one master with up to eight slaves to handle up to 64 priority levels.

A typical system is shown in Figure 12. The master controls, through the 3 line cascade bus, which one of the slaves will release the corresponding address.

As shown in Figure 12, the slaves interrupt outputs are connected to the master interrupt request inputs. When a slave request line is activated and later acknowledged, the master releases the 8080 CALL code during the first  $\overline{INTA}$  pulse. From the trailing edge of this first  $\overline{INTA}$  pulse until the trailing edge of the third pulse, the CAS lines will contain the slave address code. Thus, the corresponding slave is enabled to release the two-byte service routine address during the second and third  $\overline{INTA}$  pulses.

Note that since the CAS lines default to 000, no slave should be connected with IR0 on the master unless all other master request inputs (IR1-IR7) are connected to slaves. Otherwise, the slave on IR0 will attempt to drive the data bus in conflict with a non-slave interrupt request on the master.

It is obvious that each 8259 in the system must follow a separate initialization sequence and can be programmed to work in a different mode. An EOI command must be issued twice: once for the master and once for the corresponding slave. An address decoder is required to activate the Chip Select ( $\overline{CS}$ ) input of each 8259. The slave program pin ( $\overline{SP}$ ) must be at a "low" level for a slave (and then the cascade lines are inputs) and at a "high" level for a master (and then the cascade lines are outputs).

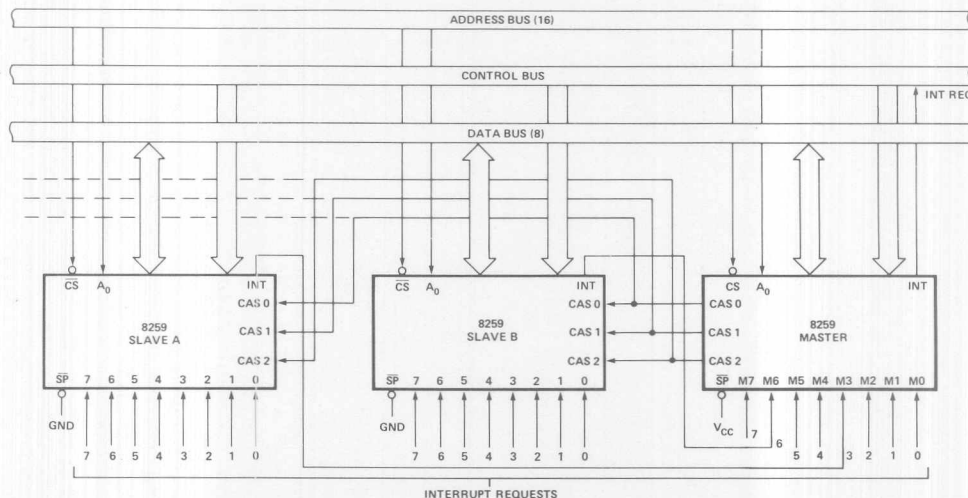


Figure 12. Cascading the 8259

INST. NO.	A0	D7	D6	D5	D4	D3	D2	D1	D0	OPERATION DESCRIPTION
1 ICW1 A	0	A7	A6	A5	1	0	1	1	0	Byte 1 initialization, format = 4, single.
2 ICW1 B	0	A7	A6	A5	1	0	1	0	0	Byte 1 initialization, format = 4, not single.
3 ICW1 C	0	A7	A6	A5	1	0	0	1	0	Byte 1 initialization, format = 8, single.
4 ICW1 D	0	A7	A6	A5	1	0	0	0	0	Byte 1 initialization, format = 8, not single.
5 ICW2	1	A15	A14	A13	A12	A11	A10	A9	A8	Byte 2 initialization (Address No. 2)
6 ICW3 M	1	S7	S6	S5	S4	S3	S2	S1	S0	Byte 3 initialization — master.
7 ICW3 S	1	0	0	0	0	0	S2	S1	S0	Byte 3 initialization — slave.
8 OCW1	1	M7	M6	M5	M4	M3	M2	M1	M0	Load mask reg, read mask reg.
9 OCW2 E	0	0	0	1	0	0	0	0	0	Non specific EOI.
10 OCW2 SE	0	0	1	1	0	0	L2	L1	L0	Specific EOI. L2, L1, L0 code of IS FF to be reset.
11 OCW2 RE	0	1	0	1	0	0	0	0	0	Rotate at EOI (Auto Mode).
12 OCW2 RSE	0	1	1	1	0	0	L2	L1	L0	Rotate at EOI (Specific Mode). L2, L1, L0, code of line to be reset and selected as bottom priority.
13 OCW2 RS	0	1	1	0	0	0	L2	L1	L0	L2, L1, L0 code of bottom priority line.
14 OCW3 P	0	—	0	0	0	1	1	0	0	Poll mode.
15 OCW3 RIS	0	—	0	0	0	1	0	1	1	Read IS register.
16 OCW3 RR	0	—	0	0	0	1	0	1	0	Read requests register.
17 OCW3 SM	0	—	1	1	0	1	0	0	0	Set special mask mode.
18 OCW3 RSM	0	—	1	0	0	1	0	0	0	Reset special mask mode.

Notes:

1. In the master mode SP pin = 1, in slave mode SP = 0.
2. (—) = do not care.

Figure 13. 8259 Instruction Set

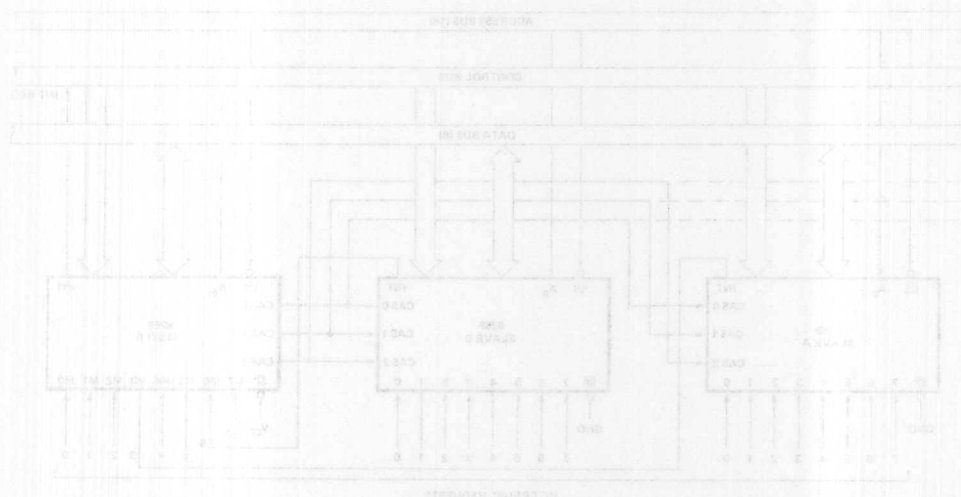


Figure 12. Cascading the 8259

## ABSOLUTE MAXIMUM RATINGS\*

Ambient Temperature Under Bias ..... 0° C to 70° C  
 Storage Temperature ..... -65° C to +150° C  
 Voltage On Any Pin  
     With Respect to Ground ..... -0.5 V to +7 V  
 Power Dissipation ..... 1 Watt

## \*COMMENT:

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

## D.C. CHARACTERISTICS

(T<sub>A</sub> = 0° C to 70° C; V<sub>CC</sub> = 5V ±5%)

SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
V <sub>IL</sub>	Input Low Voltage	-.5	.8	V	
V <sub>IH</sub>	Input High Voltage	2.0	V <sub>CC</sub> +5V	V	
V <sub>OL</sub>	Output Low Voltage		.45	V	I <sub>OL</sub> = 2 mA
V <sub>OH</sub>	Output High Voltage	2.4		V	I <sub>OH</sub> = -400 μA
V <sub>OH-INT</sub>	Interrupt Output High Voltage	2.4		V	I <sub>OH</sub> = -400 μA
		3.5		V	I <sub>OH</sub> = -50 μA
I <sub>IL</sub> (I <sub>IR0-7</sub> )	Input Leakage Current for IR <sub>0-7</sub>		-300	μA	V <sub>IN</sub> = 0V
			10	μA	V <sub>IN</sub> = V <sub>CC</sub>
I <sub>IL</sub>	Input Leakage Current for Other Inputs		10	μA	V <sub>IN</sub> = V <sub>CC</sub> to 0V
I <sub>OFL</sub>	Output Float Leakage		±10	μA	V <sub>OUT</sub> = 0.45V to V <sub>CC</sub>
I <sub>CC</sub>	V <sub>CC</sub> Supply Current		100	mA	

## CAPACITANCE

T<sub>A</sub> = 25° C; V<sub>CC</sub> = GND = 0V

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNIT	TEST CONDITIONS
C <sub>IN</sub>	Input Capacitance			10	pF	f <sub>c</sub> = 1 MHz
C <sub>I/O</sub>	I/O Capacitance			20	pF	Unmeasured pins returned to V <sub>SS</sub>

**A.C. CHARACTERISTICS**(T<sub>A</sub> = 0°C to 70°C; V<sub>CC</sub> = +5V ±5%, GND = 0V)**Bus Parameters**

Read:

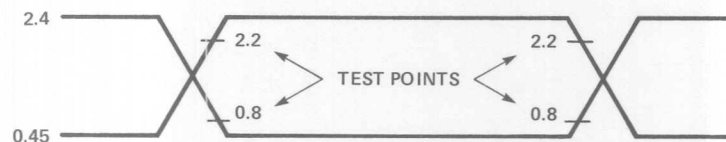
SYMBOL	PARAMETER	8259		8259-5		UNIT
		MIN.	MAX.	MIN.	MAX.	
t <sub>AR</sub>	$\overline{CS}/A_0$ Stable Before $\overline{RD}$ or $\overline{INTA}$	50		50		ns
t <sub>RA</sub>	$\overline{CS}/A_0$ Stable After $\overline{RD}$ or $\overline{INTA}$	5		30		ns
t <sub>RR</sub>	$\overline{RD}$ Pulse Width	420		300		ns
t <sub>RD</sub>	Data Valid From $\overline{RD}/\overline{INTA}$ (1)		300		200	ns
t <sub>DF</sub>	Data Float After $\overline{RD}/\overline{INTA}$	20	200	20	100	ns

Write:

SYMBOL	PARAMETER	8259		8259-5		UNIT
		MIN.	MAX.	MIN.	MAX.	
t <sub>AW</sub>	A <sub>0</sub> Stable Before $\overline{WR}$	50		50		ns
t <sub>WA</sub>	A <sub>0</sub> Stable After $\overline{WR}$	20		30		ns
t <sub>WW</sub>	$\overline{WR}$ Pulse Width	400		300		ns
t <sub>DW</sub>	Data Valid to $\overline{WR}$ (T.E.)	300		250		ns
t <sub>WD</sub>	Data Valid After $\overline{WR}$	40		30		ns

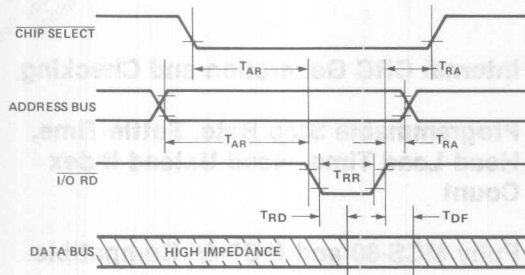
Other Timings:

SYMBOL	PARAMETER	8259		8259-5		UNIT
		MIN.	MAX.	MIN.	MAX.	
t <sub>IW</sub>	Width of Interrupt Request Pulse	100		100		ns
t <sub>INT</sub>	INT ↑ After IR ↑	400		350		ns
t <sub>IC</sub>	Cascade Line Stable After $\overline{INTA}$ ↑	400		400		ns

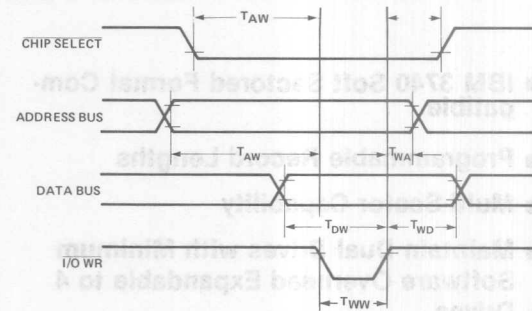
Note 1: 8259: C<sub>L</sub> = 100pF, 8259-5: C<sub>L</sub> = 150pF.**Input Waveforms for A.C. Tests**

## WAVEFORMS

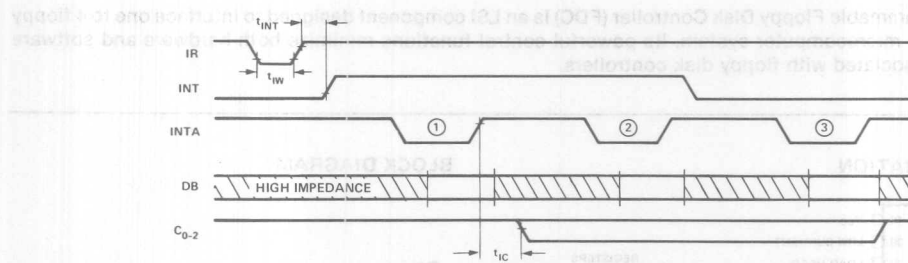
## Read Timing



## Write Timing

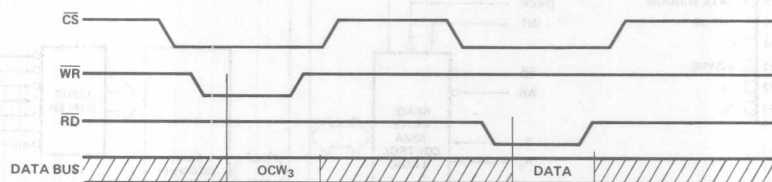


## Other Timing



Note: Interrupt Request must remain "HIGH" (at least) until leading edge of first INTA.

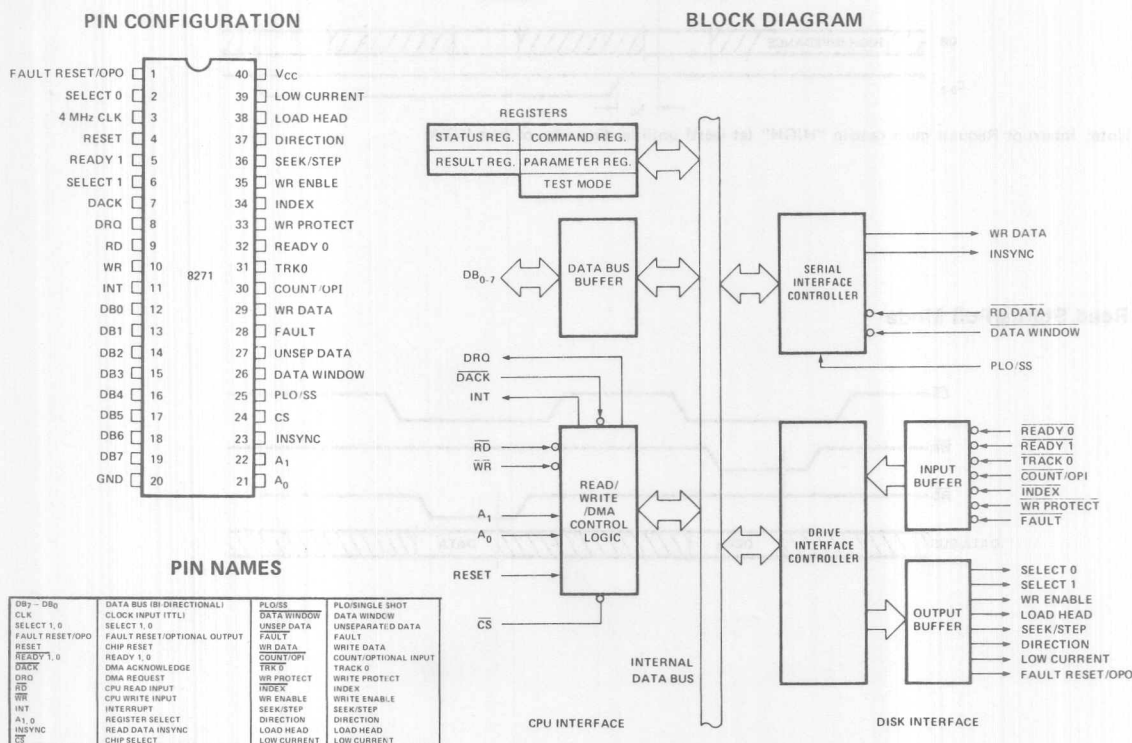
## Read Status/Poll Mode



# 8271 PROGRAMMABLE FLOPPY DISK CONTROLLER

- IBM 3740 Soft Sector Format Compatible
- Programmable Record Lengths
- Multi-Sector Capability
- Maintain Dual Drives with Minimum Software Overhead Expandable to 4 Drives
- Automatic Read/Write Head Positioning and Verification
- Internal CRC Generation and Checking
- Programmable Step Rate, Settle-Time, Head Load Time, Head Unload Index Count
- Fully MCS-80 and MCS-85 Compatible
- Single +5V Supply
- 40-Pin Package

The Intel® 8271 Programmable Floppy Disk Controller (FDC) is an LSI component designed to interface one to 4 floppy disk drives to an 8-bit microcomputer system. Its powerful control functions minimize both hardware and software overhead normally associated with floppy disk controllers.



## IBM DISKETTE GENERAL FORMAT INFORMATION

The IBM Flexible Diskette used for data storage and retrieval is organized into concentric circular paths or TRACKS. There are 77 tracks on either one or both sides (surfaces) of the diskette. On double-sided diskettes, the corresponding top and bottom tracks are referred to as a CYLINDER. Each track is further divided into fixed length sections or SECTORS. The number of sectors per track — 26, 15 or 8 — is determined when a track is formatted and is dependent on the sector length — 128, 256 or 512 bytes respectively — specified.

All tracks on the diskette are referenced to a physical index mark (a small hole in the diskette). Each time the hole passes a photodetector cell (one revolution of the diskette), an Index pulse is generated to indicate the logical beginning of a track. This index pulse is used to initiate a track formatting operation.

### Track Format

Each Diskette Surface is divided into 77 tracks with each track divided into fixed length sectors. A sector can hold a whole record or a part of a record. If the record is shorter than the sector length, the unused bytes are filled with binary zeros. If a record is longer than the sector length, the record is written over as many sectors as its length requires. The sector size that provides the most efficient

use of diskette space can be chosen depending upon the record length required.

Tracks are numbered from 00 (outer-most) to 76 (inner-most) and are used as follows:

TRACK 00 reserved as System Label Track

TRACKS 01 through 74 used for data

TRACKS 75 and 76 used as alternates.

Each sector consists of an ID field (which holds a unique address for the sector) and a data field.

The ID field is seven bytes long and is written for each sector when the track is formatted. Each ID field consists of an ID field Address Mark, a Cylinder Number byte which identifies the track number, a Head Number byte which specifies the head used (top or bottom) to access the sector, a Record Number byte identifying the sector number (1 through 26 for 128 byte sectors on standard drives), an N-byte specifying the byte length of the sector and two CRC (Cyclic Redundancy Check) bytes.

The Gaps separating the index mark and the ID and data fields are written on a track when it is formatted. These gaps provide both an interval for switching the drive electronics from reading or writing and compensation for rotational speed and other diskette-to-diskette and drive-to-drive manufacturing tolerances to ensure that data written on a diskette by one system can be read by another (diskette interchangeability).

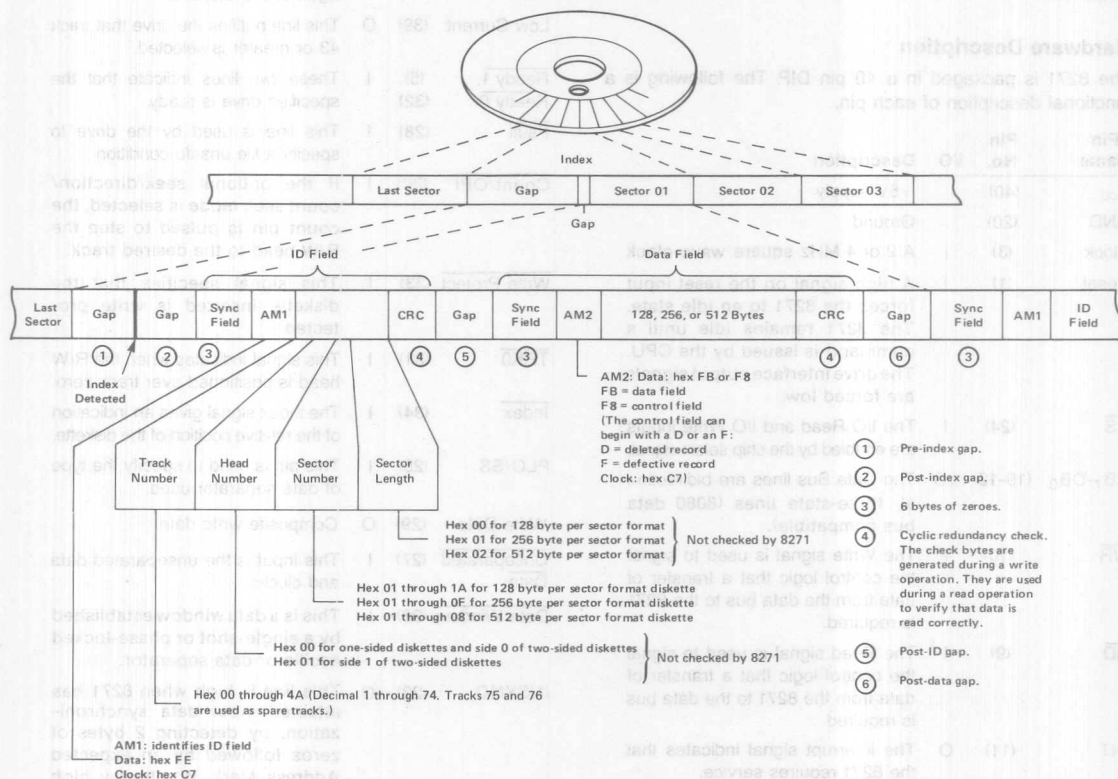


FIGURE 1. TRACK FORMAT

## General

The 8271 Floppy Disk Controller (FDC) interfaces either two single or one dual floppy drive to an eight bit microprocessor and is fully compatible with Intel's new high performance MCS-85 microcomputer system. With minimum external circuitry, this innovative controller supports most standard, commonly-available flexible disk drives including the mini-floppy.

The 8271 FDC supports a comprehensive soft sector format which is IBM 3740 compatible and includes provision for the designating and handling of bad tracks. It is a high level controller that relieves the CPU (and user) of many of the control tasks associated with implementing a floppy disk interface. The FDC supports a variety of high level instructions which allow the user to store and retrieve data on a floppy disk without dealing with the low level details of disk operation.

In addition to the standard read/write commands, a scan command is supported. The scan command allows the user program to specify a data pattern and instructs the FDC to search for that pattern on a track. Any application that is required to search the disk for information (such as point of sale price lookup, disk directory search, etc.), may use the scan command to reduce the CPU overhead. Once the scan operation is initiated, no CPU intervention is required.

## Hardware Description

The 8271 is packaged in a 40 pin DIP. The following is a functional description of each pin.

Pin Name	Pin No.	I/O	Description
V <sub>CC</sub>	(40)		+5V supply
GND	(20)		Ground
Clock	(3)	I	A 2 or 4 MHz square wave clock
Reset	(1)	I	A high signal on the reset input forces the 8271 to an idle state. The 8271 remains idle until a command is issued by the CPU. The drive interface output signals are forced low.
$\overline{CS}$	(24)	I	The I/O Read and I/O Write inputs are enabled by the chip select signal.
DB <sub>7</sub> -DB <sub>0</sub>	(19-12)	I/O	The Data Bus lines are bidirectional, three-state lines (8080 data bus compatible).
$\overline{WR}$	(10)	I	The Write signal is used to signal the control logic that a transfer of data from the data bus to the 8271 is required.
$\overline{RD}$	(9)	I	The Read signal is used to signal the control logic that a transfer of data from the 8271 to the data bus is required.
INT	(11)	O	The interrupt signal indicates that the 8271 requires service.
A <sub>1</sub> -A <sub>0</sub>	(22-21)	I	These two lines are CPU Interface Register select lines.

Name	No.	I/O	Description
DRQ	(8)	O	The DMA request signal is used to request a transfer of data between the 8271 and memory.
$\overline{DACK}$	(7)	I	The DMA acknowledge signal notifies the 8271 that a DMA cycle has been granted.
Select 1- Select 0	(6) (2)	O	These lines are used to specify the selected drive.
Fault Reset/ OPO	(1)	O	The optional fault reset output line is used to reset an error condition which is latched by the drive.
Write Enable	(35)	O	This signal enables the drive write logic.
Seek/Step	(36)	O	This multi-function line is used during drive seeks.
Direction	(37)	O	The direction line specifies the seek direction. A high level on this pin steps the R/W head toward the spindle (step-in), a low level steps the head away from the spindle (step-out).
Load Head	(38)	O	The load head line causes the drive to load the Read/Write head against the diskette.
Low Current	(39)	O	This line notifies the drive that track 43 or greater is selected.
Ready 1, Ready 0	(5) (32)	I	These two lines indicate that the specified drive is ready.
Fault	(28)	I	This line is used by the drive to specify a file unsafe condition.
Count/OPI	(30)	I	If the optional seek/direction/count seek mode is selected, the count pin is pulsed to step the R/W head to the desired track.
Write Protect	(33)	I	This signal specifies that the diskette inserted is write protected.
TRK0	(31)	I	This signal indicates when the R/W head is positioned over track zero.
Index	(34)	I	The index signal gives an indication of the relative position of the diskette.
PLO/SS	(25)	I	This pin is used to specify the type of data separator used.
Write Data	(29)	O	Composite write data.
Unseparated Data	(27)	I	This input is the unseparated data and clocks.
Data Window	(26)	I	This is a data window established by a single-shot or phase-locked oscillator data separator.
INSYNC	(23)	O	This line is high when 8271 has attained input data synchronization, by detecting 2 bytes of zeros followed by an expected Address Mark. It will stay high until the end of the ID or data field.

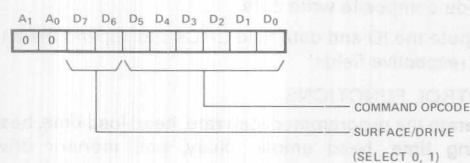
## CPU Interface Description

This interface minimizes CPU involvement by supporting a set of high level commands and both DMA and non-DMA type data transfers and by providing hierarchical status information regarding the result of command execution.

The CPU utilizes the control interface (see the Block diagram) to specify the FDC commands and to determine the result of an executed command. This interface is supported by five Registers which are addressed by the CPU via the A<sub>1</sub>, A<sub>0</sub>,  $\overline{RD}$  and  $\overline{WR}$  signals. If an 8080 based system is used, the  $\overline{RD}$  and  $\overline{WR}$  signals can be driven by the 8228's  $\overline{I/OR}$  and  $\overline{I/OW}$  signals. The registers are defined as follows:

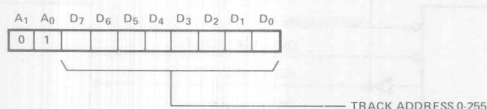
### Command Register

The CPU loads an appropriate command into the Command Register which has the following format:



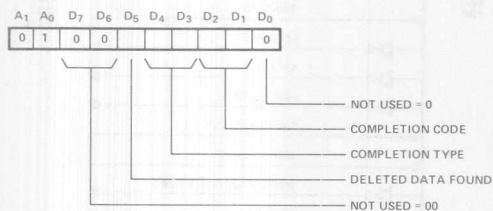
### Parameter Register

Accepts parameters of commands that require further description; up to five parameters may be required, example:



### Result Register

The Result Register is used to supply the outcome of FDC command execution (such as a good/bad completion) to the CPU. The standard Result byte format is:



This byte format facilitates the use of an address table to look up error routines and messages.

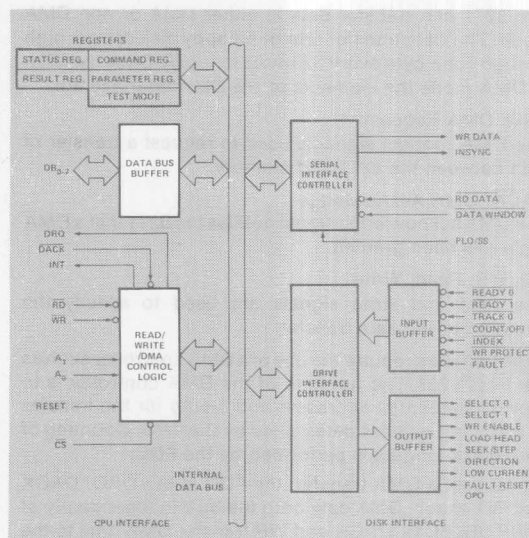
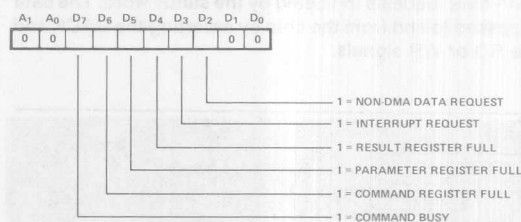


FIGURE 2. 8271 BLOCK DIAGRAM SHOWING CPU INTERFACE FUNCTIONS

### Status Register

Reflects the state of the FDC.



### Test Mode

Allows the 8271 to be reset by the program.

### INT (Interrupt Line)

Another element of the control interface is the Interrupt line (INT). This line is used to signal the CPU that an FDC operation has been completed. It remains active until the result register is read.

### DMA Operation

The 8271 can transfer data in either DMA or non DMA mode. The data transfer rate of a floppy disk drive is high enough (one byte every 32 usec) to justify DMA transfer. In DMA mode the elements of the DMA interface are:

#### DRQ: DMA Request:

The DMA request signal is used to request a transfer of data between the 8271 and memory.

#### DACK: DMA Acknowledge:

The DMA acknowledge signal notifies the 8271 that a DMA cycle has been granted.

#### RD, WR: Read, Write

The read and write signals are used to specify the direction of the data transfer.

DMA transfers require the use of a DMA controller such as the Intel® 8257. The function of the DMA controller is to provide sequential addresses and timing for the transfer at a starting address determined by the CPU. Counting of data block lengths is performed by the FDC.

To request a DMA transfer, the FDC raises DRQ. DACK and RD enable DMA data onto the bus (independently of CHIP SELECT). DACK and WR transfer DMA data to the FDC. If a data transfer request (read or write) is not serviced within 31  $\mu$ sec, the command is cancelled, a late DMA status is set, and an interrupt is generated. In DMA mode, an interrupt is generated at the completion of the data block transfer.

When configured to transfer data in non-DMA mode, the CPU must pass data to the FDC in response to the non-DMA data requests indicated by the status word. The data is passed to and from the chip by asserting the DACK and the RD or WR signals.

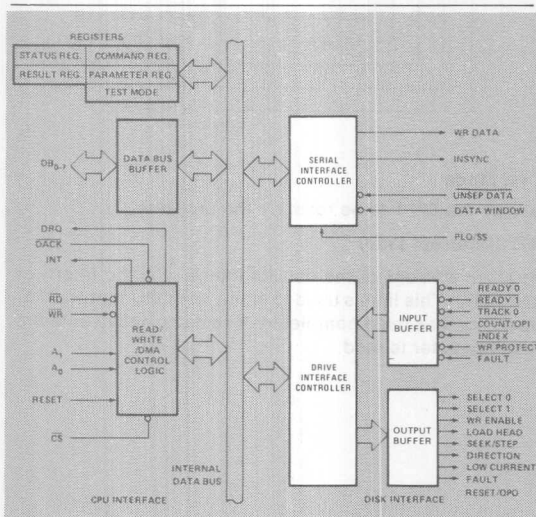


FIGURE 3. 8271 BLOCK DIAGRAM SHOWING DISK INTERFACE FUNCTIONS

### Disk Drive Interface

The 8271 disk drive interface supports the high level command structure described in the Command Description section. The 8271 maintains the location of bad tracks and the current track location for two drives. However, with minor software support, this interface can support four drives by expanding the two drive select lines (select 0, select 1) with the addition of minimal support hardware.

The FDC Disk Drive Interface has the following major functions.

#### READ FUNCTIONS

Utilize the user supplied data window to obtain the clock and data patterns from the unseparated read data.

Establish byte synchronization.

Compute and verify the ID and data field CRCs.

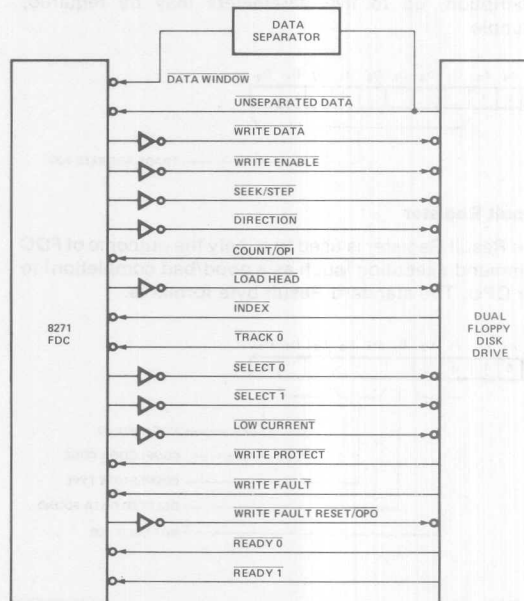
#### WRITE FUNCTIONS

Encode composite write data.

Compute the ID and data field CRCs and append them to their respective fields.

#### CONTROL FUNCTIONS

Generate the programmed step rate, head load time, head settling time, head unload delay, and monitor drive functions.



NOTE: INPUTS TO CHIP MAY REQUIRE RECEIVERS (AT LEAST PULL UP/DOWN PAIRS).

FIGURE 4. 8271 DISK DRIVE INTERFACE

### Data Separation

The 8271 needs only a data window to separate the data from the composite read data as well as to detect missing clocks in the Address Marks.

The window generation logic may be implemented using either a single-shot separator or a phase-locked oscillator.

### Single-Shot Separator

The single-shot separator approach is the lowest cost solution.

The FDC samples the value of Data Window on the leading edge of Unseparated Data and determines whether the delay from the previous pulse was a half or full bit-cell (high input = full bit-cell, low input = half bit-cell).

PLO/SS should be tied to Ground.

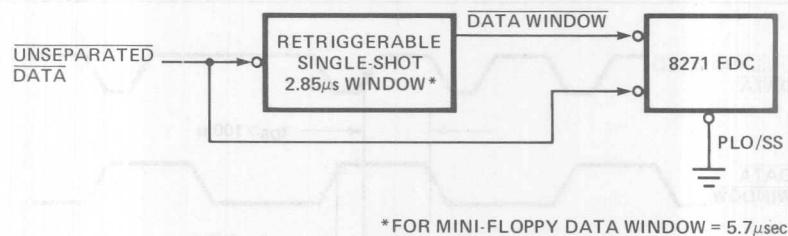


FIGURE 5. SINGLE-SHOT DATA SEPARATOR BLOCK DIAGRAM

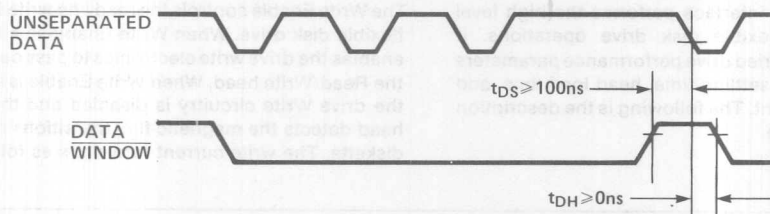


FIGURE 6. SINGLE-SHOT DATA WINDOW TIMING

The FDC samples the value of Data Window on the leading edge of Unseparated Data and determines whether the pulse represents a Clock or Data Pulse.

PLO/SS should be tied to Vcc (+5V).

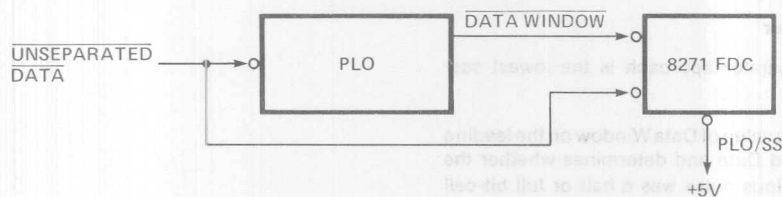


FIGURE 7. PLO DATA SEPARATOR BLOCK DIAGRAM

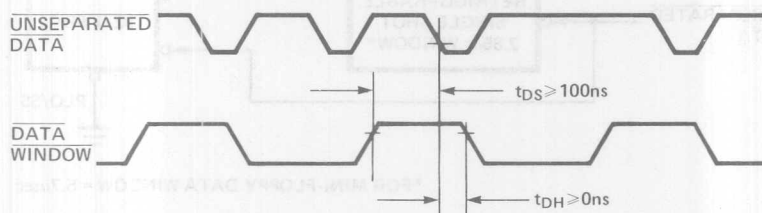


FIGURE 8. PLO DATA WINDOW TIMING

#### Disk Drive Control Interface

The disk drive control interface performs the high level and programmable flexible disk drive operations. It custom tailors many varied drive performance parameters such as the step rate, settling time, head load time, and head unload index count. The following is the description of the control interface.

#### Write Enable

The Write Enable controls the read and write functions of a flexible disk drive. When Write Enable is a logical one, it enables the drive write electronics to pass current through the Read/Write head. When Write Enable is a logical zero, the drive Write circuitry is disabled and the Read/Write head detects the magnetic flux transitions recorded on a diskette. The write current turn-on is as follows.

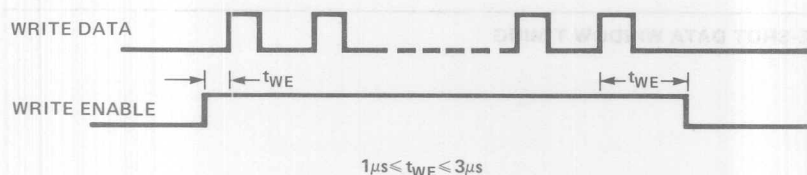


FIGURE 9. WRITE ENABLE TIMING

### Seek Control

Seek Control is accomplished by Seek/Step, Direction, and Count pins and can be implemented two ways to provide maximum flexibility in the subsystem design. One instance is when the programmed step rate is not equal to zero. In this case, the 8271 uses the Seek/Step and Direction pins (the Seek/Step pin becomes a Step pin). Programmable Step timing parameters are shown.

Another instance is when the programmable step rate is equal to zero, in which case the 8271 holds the seek line high until the appropriate number of user-supplied step pulses have been counted on the count input pin.

The Direction pin is a control level indicating the direction in which the R/W head is stepped. A logic high level on this line moves the head toward the spindle (step-in). A logic low level moves the head away from the spindle (step-out).

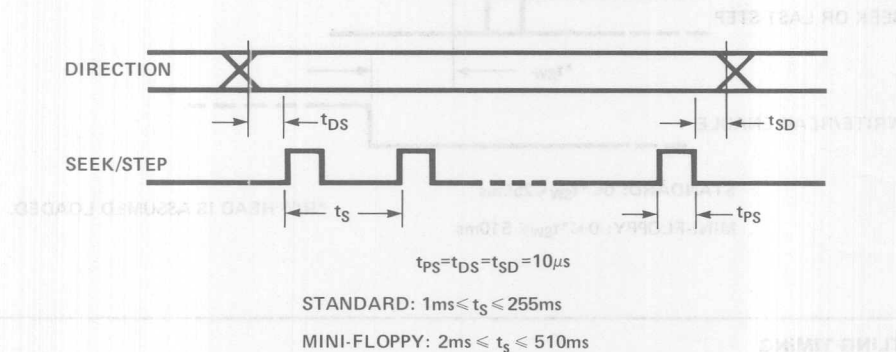


FIGURE 10. SEEK TIMING

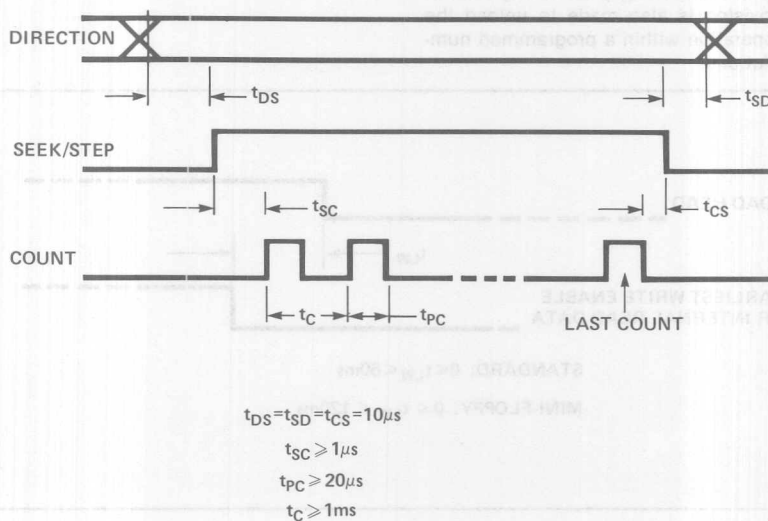
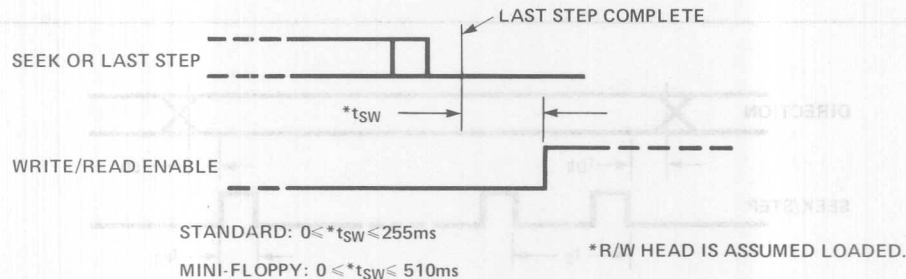


FIGURE 11. SEEK/STEP/COUNT TIMING

### Head Settling Time

The 8271 allows the head settling time to be programmed from 0 to 255ms, in increments of 1 ms for standard drives or 0 to 510ms for mini-drives.

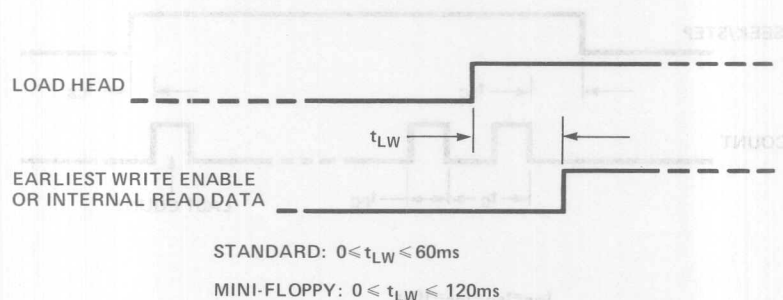
The head settling time is defined as the interval of time from completion of the last step to the time when reading or writing on the diskette is possible (R/W Enable). The R/W head is assumed loaded.



### HEAD SETTTLING TIMING

#### Load Head

When active, load head output pin causes the drive's read/write head to be loaded on the diskette. When the head is initially loaded, there is a programmed delay (0 to 60ms in 4ms increments for standard drives or 0 to 120ms in 8 ms increments for mini-drives) prior to any read or write operation. Provision is also made to unload the head following an operation within a programmed number of diskette revolutions.



### HEAD LOAD TO READ/WRITE TIMING

## Index

The Index input is used to determine "Sector not found" status and to initiate format track/read ID commands and head unload Index and Count operations.

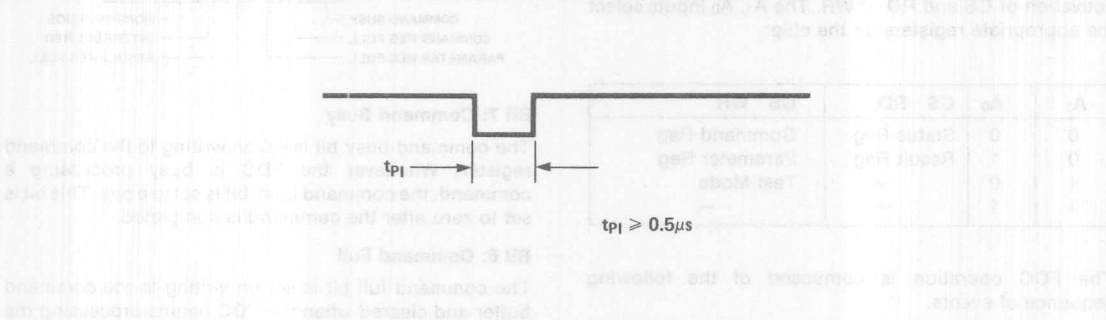


FIGURE 14. INDEX TIMING

### Track 0

This input pin indicates that the diskette is at track 0. During any seek operation, the stepping out of the actuator ceases when the track 0 pin becomes active.

### Select 0, Select 1

Only one drive may be selected at a time. The Input/Output pins that must be externally qualified with Select 0 and Select 1 are:

- Unseparated Data
- Data Window
- Write Enable
- Seek/Step
- Count/Optional Input
- Load Head
- Track 0
- Low Current
- Write Protect
- Write Fault
- Fault Reset/Optional Output
- Index

When a new set of select bits is specified by a new command or the FDC going IDLE, the following pins will be set to the 0 state:

- Write Enable (35)
- Seek/Step (36)
- Direction (37)
- Load Head (38)
- Low Head Current (39)

The select pins will be set to the state specified by the command or the FDC IDLE condition.

### Low Current

This output pin is active whenever the physical track location of the selected drive is greater than 43. Generally this signal is used to enable compensation for the lower velocities encountered while recording on the inner tracks.

### Write Protect

The 8271 will not write to a disk when this input pin is active and will interrupt the CPU if a Write attempt is made. Operations which check Write Protect are aborted if the Write Protect line is active.

This signal normally originates from a sensor which detects the presence or absence of the Write Protect hold in the diskette jacket.

### Write Fault and Write Fault Reset

The Write Fault input is normally latched by the drive and indicates any condition which could endanger data integrity. The 8271 interrupts the CPU anytime Write Fault is detected during an operation and immediately resets the Write Enable, Seek/Step, Direction, and Low Current signals. The Write Fault condition is reset through the Write Fault Reset output pin.

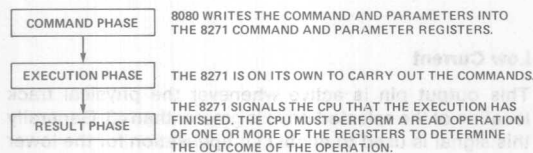
### Ready 1, 0

These two pins indicate the functional status of the disk drives. Whenever an operation is attempted on a drive which is not ready, an interrupt is generated. The interface continually monitors this input during an operation and if a Not Ready condition occurs, immediately terminates the operation. Note that the 8271 latches the Not Ready condition and it can only be reset by the execution of a Read Drive Status command.

from the CPU, executes them and provides a RESULT back to the 8080 CPU at the end of command execution. The communication with the CPU is established by the activation of CS and RD or WR. The A<sub>1</sub>, A<sub>0</sub> inputs select the appropriate registers on the chip:

A <sub>1</sub>	A <sub>0</sub>	CS RD	CS WR
0	0	Status Reg	Command Reg
0	1	Result Reg	Parameter Reg
1	0	—	Test Mode
1	1	—	—

The FDC operation is composed of the following sequence of events.



### The Command Phase

The software writes a command to the command register. As a function of the command issued, from zero to five parameters are written to the parameter register. Refer to diagram showing a flow chart of the command phase. Note that the flow chart shows that a command may not be issued if the FDC status register indicates that the device is busy. Issuing a command while another command is in progress is illegal. The flow chart also shows a parameter buffer full check. The FDC status indicates the state of the parameter buffer. If a parameter is issued while the parameter buffer is full, the previous parameter is over written and lost.

### The Execution Phase

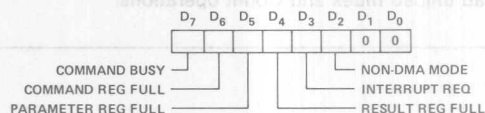
During the execution phase the operation specified during the command phase is performed. During this phase, there is no CPU involvement if the system utilizes DMA for the data transfers. The execution phase of each command is discussed within the detailed command descriptions. The following table summarizes many of the basic execution phase characteristics.

### The Result Phase

During the Result Phase, the FDC notifies the CPU of the outcome of the command execution. This phase may be initiated by:

1. The successful completion of an operation.
2. An error detected during an operation.

in the Result Phase, the CPU Reads the Status Register which provides the following information:



### Bit 7: Command Busy

The command busy bit is set on writing to the command register. Whenever the FDC is busy processing a command, the command busy bit is set to a one. This bit is set to zero after the command is completed.

### Bit 6: Command Full

The command full bit is set on writing to the command buffer and cleared when the FDC begins processing the command.

### Bit 5: Parameter Full

This bit indicates the state of the parameter buffer. This bit is set when a parameter is written to the FDC and reset after the FDC has accepted the parameter.

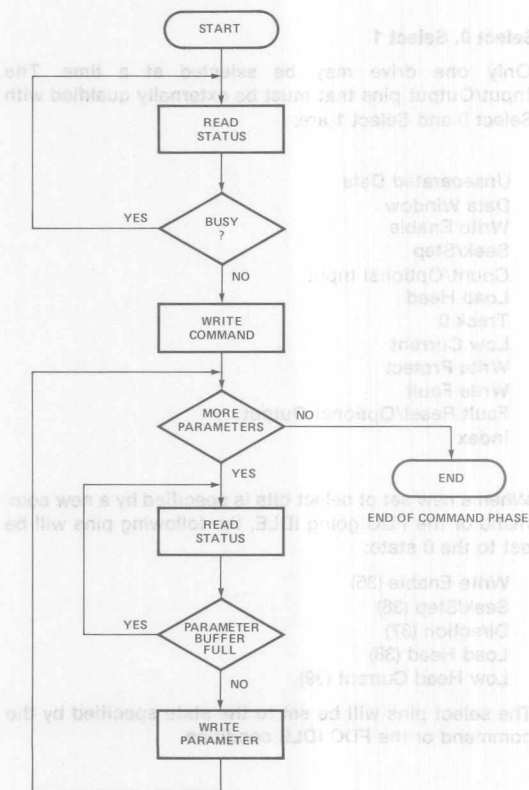


FIGURE 5. COMMAND PHASE SEQUENCE

**Bit 4: Result Full**

This bit indicates the state of the result buffer. It is valid only after Command Busy bit is low. This bit is set when the FDC begins a command and is reset after the result byte is read by the CPU. The data in the result buffer is valid only after the FDC has completed a command. Reading the result buffer while a command is in progress yields no useful information.

**Bit 3: Interrupt Request**

This bit reflects the state of the FDC INT pin. It is set where FDC requests attention as a result of the completion of an operation or failure to complete an intended operation. This bit is cleared by a read result register interface command.

**Bit 2: Non-DMA Data Request**

When the FDC is utilized without a DMA controller, this bit is used to indicate FDC data requests. Note that in the non-DMA mode, an interrupt is generated (interrupt request bit is set) with each data byte written to or read from the diskette.

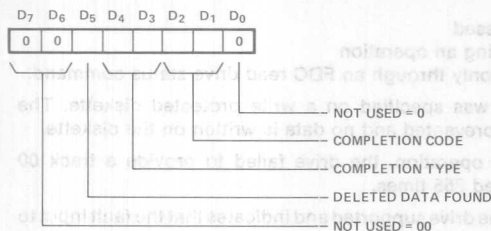
**Bits 1 and 0:**

Not used (zero returned).

After reading the Status Register, the CPU then reads the Result Register for more information.

**The Result Register**

The standard result byte format is:

**Bits 7 and 6:**

Not used (zero returned).

**Bit 5:**

**Deleted Data Found:** This bit is set when deleted data is encountered during a transaction.

**Bits 4 and 3: Completion Type**

The completion type field provides general information regarding the outcome of an operation.

The completion type field provides general information regarding the outcome of an operation.

Completion Type	Event
00	Good Completion — No Error
01	System Error — recoverable errors;
10	operator intervention probably required for recovery.
11	Command/Drive Error — either a program error or drive hardware failure.

**Bits 2 and 1: Completion Code**

The completion code field provides more detailed information about the completion type (See Table).

Completion Type	Completion Code	Event
00	00	Good Completion/
00	01	Scan Not Met
00	10	Scan Met Equal
00	11	Scan Met Not Equal
01	00	—
01	01	Clock Error
01	10	Late DMA
01	11	ID CRC Error
10	00	Data CRC Error
10	01	Drive Not Ready
10	10	Write Protect
10	11	Track 0 Not Found
11	00	Write Fault
11	01	Sector Not Found
11	10	—
11	11	—

It is important to note the hierarchical structure of the result byte. In very simple systems where only a GO-NO GO result is required, the user may simply branch on a zero result (a zero result is a good completion). The next level of complexity is at the completion type interface. The completion type supplies enough information so that the software may distinguish between fatal and non-fatal errors. If a completion type 01 occurs, ten retries should be performed before the error is considered unrecoverable.

The Completion Type/Completion Code interface supplies the greatest detail about each type of completion. This interface is used when detailed information about the transaction completion is required.

**Bit 0:**

Not used (zero returned).

TABLE 1. COMPLETION CODE INTERPRETATION

Definition	Interpretation
Successful Completion/ Scan Not Met	The diskette operation specified was completed without error. If scan operation was specified, the pattern scanned was not found on the track addressed.
Scan Met Equal	The data pattern specified with the scan command was found on the track addressed with the specified comparison, and the equality was met.
Scan Met Not Equal	The data pattern specified with the scan command was found with the specified comparison on the track addressed, but the equality was not met.
Clock Error	During a diskette read operation, a clock bit was missing (dropped). Note that this function is disabled when reading any of the ID address marks (which contain missing clock pulses). If this error occurs, the operation is terminated immediately and an interrupt is generated.
Late DMA	During either a diskette read or write operation, the data channel did not respond within the allotted time interval to prevent data from being overwritten or lost. This error immediately terminates the operation and generates an interrupt.
ID Field CRC Error	The CRC word (two bytes) derived from the data read in an ID field did not match the CRC word written in the ID field when the track was formatted. If this error occurs, the associated diskette operation is prevented and no data is transferred.
Data Field CRC Error	During a diskette read operation, the CRC word derived from the data field read did not match the data field CRC word previously written. If this error occurs, the data read from the sector should be considered invalid.
Drive Not Ready	The drive addressed was not ready. This indication is caused by any of the following conditions: <ol style="list-style-type: none"> <li>1. Drive not powered up</li> <li>2. Diskette not loaded</li> <li>3. Non-existent drive addressed</li> <li>4. Drive went not ready during an operation</li> </ol> Note that this bit is cleared only through an FDC read drive status command.
Write Protect	A diskette write operation was specified on a write protected diskette. The intended write operation is prevented and no data is written on the diskette.
Track 00 Not Found	During a seek to track 00 operation, the drive failed to provide a track 00 indication after being stepped 255 times.
Write Fault	This error is dependent on the drive supported and indicates that the fault input to the FDC has been activated by the drive.
Sector Not Found	Either the sector addressed could not be found within one complete revolution of the diskette (two index marks encountered) or the track address specified did not match the track address contained in the ID field. Note that when the track address specified and the track address read do not match, the FDC automatically increments its track address register (stepping the drive to the next track) and again compares the track addresses. If the track addresses still do not match, the track address register is incremented a second time and another comparison is made before the sector not found bit is set.

## EXECUTION PHASE BASIC CHARACTERISTICS

The following table summarizes the various commands with corresponding execution phase characteristics.

COMMANDS	1 Deleted Data	2 Head	3 Ready	4 Write/ Protect	5 Seek	6 Seek Check	7 Result	8 Completion Interrupt
SCAN DATA	SKIP	LOAD	✓	x	YES	YES	YES	YES
SCAN DATA AND DEL DATA	XFER	LOAD	✓	x	YES	YES	YES	YES
WRITE DATA	x	LOAD	✓	✓	YES	YES	YES	YES
WRITE DEL DATA	x	LOAD	✓	✓	YES	YES	YES	YES
READ DATA	SKIP	LOAD	✓	x	YES	YES	YES	YES
READ DATA AND DEL DATA	XFER	LOAD	✓	x	YES	YES	YES	YES
READ ID	x	LOAD	✓	x	YES	NO	YES	YES
VERIFY DATA AND DEL DATA	XFER	LOAD	✓	x	YES	YES	YES	YES
FORMAT TRACK	x	LOAD	✓	✓	YES	NO	YES	YES
SEEK	x	LOAD	y	x	YES	NO	YES	YES
READ DRIVE STATUS	x	—	x	x	NO	NO	NOTE 5	NO
SPECIFY	x	—	x	x	NO	NO	NO	NO
RESET	x	UNLOAD	x	x	NO	NO	NO	NO
W SP REGISTERS	x	—	x	x	NO	NO	NO	NO
R SP REGISTERS	x	—	x	x	NO	NO	NOTE 6	NO

Note: 1. "x" → DON'T CARE 2. "✓" → check 3. "—" → No change 4. "y" → Check at end of operation 5. See "Read Drive Status" Command.

6. See "Read Special Register" Command.

TABLE 1. EXECUTION PHASE BASIC CHARACTERISTICS

Explanation of the execution phase characteristics table.

## 1. Deleted Data Processing

If deleted data is encountered during an operation that is marked skip in the table, the deleted data sector is not transferred into memory, but the sector is counted. For example, if the command and parameters specify a read of five sectors and one of the sectors was written with a deleted data mark, four sectors are transferred to memory. The deleted data flag is set in the result byte. However, if the operation is marked transfer, all data is transferred to memory regardless of the type of data mark.

## 2. Head

The Head column in the table specifies whether the Read/Write head will be loaded or not. If the table specifies load, the head is loaded after it is positioned over the track. The head loaded by a command remains loaded until the user specified number of index pulses have occurred.

## 3. Ready

The Ready column indicates if the ready line (Ready 1, Ready 0) associated with the selected drive is checked. A not ready state is latched by the 8271 until the user executes a read status command.

## 4. Write Protect

The operations that are marked check Write Protect are immediately aborted if Write Protect line is active at the beginning of an operation.

## 5. Seek

Many of the 8271 commands cause a seek to the desired track. A current track register is maintained for each drive or surface.

## 6. Seek Check

Operations that perform Seek Check verify that selected data in the ID field is correct before the 8271 accesses the data field.

## DETAILED COMMAND DESCRIPTION

Many of the interface characteristics of the FDC are specified by the systems software. Prior to initiating any drive operation command, the software must execute the specify command. There are two types of specify commands selectable by the parameter issued.

### First Parameter Specify Type

0DH	Initialization
10H	Load bad Tracks Surface '0'
18H	Load bad Tracks Surface '1'

### Specify Command

The Specify command is used prior to performing any diskette operation (including formatting of a diskette) to define the drive's inherent operating characteristics and also is used following a formatting operation or installation of another diskette to define the locations of bad tracks. Since the Specify command only loads internal registers within the 8271 and does not involve an actual diskette operation, command processing is limited to only Command Phase. Note that once the operating characteristics and bad tracks have been specified for a given drive and diskette, redefining these values need only be done if a diskette with unique bad tracks is to be used or if the system is powered down.

#### Initialization:

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	0	0	1	1	0	1	0	1
PAR0:	0	1	0	0	0	0	1	1	0	1
PAR1:	0	1	STEP RATE*							
PAR2:	0	1	HEAD SETTLING TIME*							
PAR3:	0	1	INDEX CNT BEFORE HEAD UNLOAD				HEAD LOAD TIME*			

\*Note: Mini-floppy times are doubled.

Parameter 0: 0D<sub>H</sub> = Select Specify Initialization.

Parameter 1: D<sub>7</sub>-D<sub>0</sub> = Step Rate (0-255ms in 1ms steps).  
{0-510ms in 2ms steps} ( ) = standard, { } = mini

Parameter 2: D<sub>7</sub>-D<sub>4</sub> = Head Settling Time (0-255 ms in 1 ms steps). {0-510ms in 2ms steps} ( ) = standard, { } = mini

Parameter 3: D<sub>7</sub>-D<sub>4</sub> = Index Count—Specifies the number of Revolutions (0-14) which are to occur before the FDC automatically unloads the R/W head. If 15 is specified, the head remains loaded.

D<sub>3</sub>-D<sub>0</sub> = Head Load Time (0-60ms in steps of 4 ms).  
{0-120ms in 8 ms steps} ( ) = standard, { } = mini

#### Load Bad Tracks

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	0	0	1	1	0	1	0	1
PAR0:	0	1	0	0	0	1	1/0	0	0	0
PAR1:	0	1	BAD TRACK NO. 1							
PAR2:	0	1	BAD TRACK NO. 2							
PAR3:	0	1	CURRENT TRACK							

Parameter 0: 10H = Load Surface zero bad tracks  
18H = Load Surface one bad tracks

Parameter 1:  
Bad track address number 1 (Physical Address).

Parameter 2:  
Bad track address number 2 (Physical Address).

Parameter 3:  
Current track address (Physical Address).

#### Conditions:

1. Bad track number one must be numerically less than bad track number two.
2. If no bad tracks are present, set the parameter to FF<sub>H</sub>.

#### Reset Command

A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
1	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0

Function: The Reset command emulates the action of the reset pin. It is issued by outputting a one followed by a zero to the Test Mode Register.

1. The drive control signals are forced low.
2. An in-progress command is aborted.
3. The FDC status register flags are cleared.
4. The FDC enters an idle state until the next command is issued.

## Special Drive Commands

The special drive commands are used to explicitly position the drive read/write head or to interrogate the drive status.

### Command Seek Function

The seek command moves the head to the specified track without loading the head or verifying the track.

The seek operation uses the specified bad tracks to compute the physical track address. This feature insures that the seek operation positions the head over the correct track.

When a seek to track zero is specified (bad track registers are not used), the FDC steps the head until the track 00 signal is detected.

If the track 00 signal is not detected within (FF)<sub>H</sub> steps, a track 0 not found error status is returned.

A seek to track zero is used to position the read/write head when the current head position is unknown (such as after a power up).

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	SEL 1	SEL 0	1	0	1	0	0	1
PAR0:	0	1	TRACK ADDRESS 0-255							

Seek operations are not verified. A subsequent read or write operation must be performed to determine if the correct track is located.

### Read Drive Status Command

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	SEL 1	SEL 0	1	0	1	1	0	0

RESULT: EACH BIT INDICATES CURRENT STATE OF INPUT PINS.

A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
0	1	1	RDY 1	WR FAULT	INDEX	WR PROT	RDY 0	TRACK CNT 0	OPI

IF A DRIVE NOT READY RESULT IS RETURNED, THE READ STATUS MUST BE ISSUED TO CLEAR THE CONDITION.

## Read/Write Special Register Commands

This command is used to access special registers within the 8271.

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	SEL 1	SEL 0	COMMAND OFCODE					
PAR0:	0	1	REGISTER ADDRESS							

Command code:

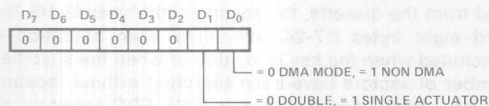
3D<sub>H</sub> Read Special Register

3A<sub>H</sub> Write Special Register

For both commands, the first parameter is the register address; for Write commands a second parameter specifies data to be written. Only the Read Special Register command supplies a result.

Description	Register Address in Hex	Comment
Sector Number	06	See Scan Description
MSB of Count	14	See Scan Description
LSB of Count	13	See Scan Description
Surface 0 Current Track	12	
Surface 1 Current Track	1A	
Mode Register	17	See Mode Register Description
Drive Control Output Port	23	See Drive Output Port Description
Drive Control Input Port	22	See Read Drive Status Description
Surface 0 Bad Track 1	10	
Surface 0 Bad Track 2	11	
Surface 1 Bad Track 1	18	
Surface 1 Bad Track 2	19	

## Mode Register Write Parameter Format



## Bits 7-2

(Not used). Must be set to zero.

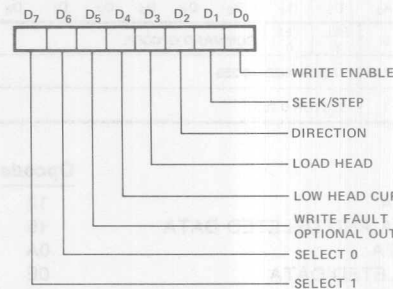
## Bit 1

Double/Single Actuator: Selects single or double actuator mode. If the single actuator mode is selected, the FDC assumes that the physical track location of both disks is always the same. This mode facilitates control of a drive which has a single actuator mechanism to move two heads.

## Bit 0

Data Transfer Mode: This bit selects the data transfer mode. If this bit is a zero, the FDC operates in the DMA mode (DMA Request/ACK). If this bit is a one, the FDC operates in non-DMA mode. When the FDC is operating in DMA mode, interrupts are generated at the completion of commands. If the non-DMA mode is selected, the FDC generates an interrupt for every data byte transferred.

## Drive Control Output Port Format



Each of these signals correspond to the chip pin of the same name. On standard-sized drives with write fault detection logic, bit 5 is set to generate the write fault reset signal. This signal is used to clear a write fault indication within the drive. On mini-sized drives, this bit is set to turn on the drive motor prior to initiating a drive operation or is cleared to turn off the drive motor following an operation. The register must be read prior to writing the register in order to save the states of the remaining bits. When the register is subsequently written to modify bit 5, the remaining bits must be restored to their previous states.

## Data Processing Commands

All the routine Read/Write commands examine specific drive status lines before beginning execution, perform an implicit seek to the track address and load the drive's read/write head. Regardless of the type of command (i.e., read, write or verify), the 8271 first reads the ID field(s) to verify that the correct track has been located and also to locate the addressed sector. When a transfer is complete (or cannot be completed), the 8271 sets the interrupt request bit in the status register and provides an indication of the outcome of the operation in the result register.

If CRC error is detected during a multisector transfer, processing is terminated with the sector in error. The address of the failing sector number can be determined by examining the Sector Number register using the Read Special Register command.

Full power of the multisector read/write commands can be realized by doing DMA transfer using Intel® 8257 DMA Controller. For example, in a 128 byte per sector multisector write command, the entire data block (containing 128 bytes times the number of sectors) can be located in a disk memory buffer. Upon completion of the command phase, the 8271 begins execution by accessing the desired track, verifying the ID field, and locating the data field of the first sector to be written. The 8271 then DMA-accesses the first sector and starts counting and writing one byte at a time until all 128 bytes are written. It then locates the data field of the next sector and repeats the procedure until all the specified sectors have been written. Upon completion of the execution phase the 8271 enters into the result phase and interrupts the CPU for availability of status and completion results. Note that all read/write commands, single or multisector are executed without CPU intervention.

## 128 Byte Single Sector Format

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	SEL 1	SEL 0	COMMAND OPCODE					
PAR0:	0	1	TRACK ADDR 0-255							
PAR1:	0	1	SECTOR 0-255							

## Commands

READ DATA  
 READ DATA AND DELETED DATA  
 WRITE DATA  
 WRITE DELETED DATA  
 VERIFY DATA AND DELETED DATA

## Opcode

12  
 16  
 0A  
 0E  
 1E

## Variable Length/Multi-Sector Format

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>						
CMD:	0	0	SEL 1	SEL 0	COMMAND OPCODE											
PAR0:	0	1														
PAR1:	0	1														
PAR2:	0	1	LENGTH				NO. OF SECTORS									

D<sub>7</sub>-D<sub>5</sub> of Parameter 2 determine the length of the disk sector.

0 0 0	128 Bytes
0 0 1	256 Bytes
0 1 0	512 Bytes
0 1 1	1024 Bytes
1 0 0	2048 Bytes
1 0 1	4096 Bytes
1 1 0	8192 Bytes
1 1 1	16,384 Bytes

## Commands

READ DATA  
 READ DATA AND DELETED DATA  
 WRITE DATA  
 WRITE DELETED DATA  
 VERIFY DATA AND DELETED DATA  
 SCAN DATA  
 SCAN DATA AND DELETED DATA

## Opcode

13  
 17  
 0B  
 0F  
 1F  
 00  
 04

## Command

Read Data, Read Data and Deleted Data.

## Function

The read command transfers data from a specified disk sector or group of sectors to memory. The operation of this command is outlined in execution phase table.

## Command

Write Data, Write Deleted Data.

## Function

The write command transfers data from memory to a specified disk sector or group of sectors.

## Commands

Verify Data and Deleted Data.

## Function

The verify command is identical to the read data and deleted data command except that the data is not transferred to memory. This command is used to check that a sector or a group of sectors has been written correctly by verifying the CRC character.

## Scan Commands

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	SEL 1	SEL 0	0	0	0	S.DATA S.DELED	0	0
PAR0:	0	1	TRACK ADDR 0-255							
PAR1:	0	1	SECTOR 0-255							
PAR2:	0	1	LENGTH				NO. OF SECTORS			
PAR3:	0	1	SCAN TYPE			STEP SIZE				
PAR4:	0	1	FIELD LENGTH (KEY)							

Command D<sub>2</sub> = 0 Scan Data  
 D<sub>2</sub> = 1 Scan Data and Deleted Data

Scan Commands, Scan Data and Scan Data and Deleted Data, are used to search a specific data pattern or "key" from memory. The 8271 FDC operation during a scan is unique in that data is read from memory and from the diskette simultaneously.

During the scan operation, the key is compared repetitively (using the 8257 DMA Controller in auto load mode) with the data read from the diskette (e.g., an eight byte key would be compared with the first eight bytes (1-8) read from the diskette, the second eight bytes (9-16), the third eight bytes (17-24), etc.). The scan operation is concluded when the key is located or when the specified number of sectors have been searched without locating the key. When concluded, the 8271 FDC requests an interrupt. The program must then read the result register to determine if the scan was successful (if the key was located). If successful, several of the FDC's special registers can be examined (read special registers command) to determine more specific information relating to the scan (i.e., the sector number in which the key was located, and the number of bytes within the sector that were not compared when the key was located).

The following factors regarding key length must be considered when establishing a key in memory.

1. When searching multiple sectors, the length of the key must be evenly divisible into the sector length to prevent the key from being split at subsequent sector boundaries. Since the character FF<sub>H</sub> is not compared, the key in memory can be padded to the required length using this character. For example, if the actual pattern compared on the diskette is twelve characters in length, the field length should be sixteen and four bytes of FF<sub>H</sub>

would be appended to the key. Consequently, the last block of sixteen bytes compared within the first sector would end at the sector boundary and the first byte of the next sector would be compared with the first byte of the key.

- Since the first byte of the key is compared with the first byte of the sector, when the pattern does not begin with the first byte of the sector, the key must be offset using the character FF<sub>H</sub>. For example, if the first byte of a nine byte pattern begins on the fifth byte of the sector, four bytes of FF<sub>H</sub> are prefixed to the key (and three bytes of FF<sub>H</sub> are appended to the key to meet the length requirement) so that the first actual comparison begins on the fifth byte.

The Scan Commands require five parameters:

#### Parameter 0, Track Address

Specifies the track number containing the sectors to be scanned. Legal values range from 00<sub>H</sub> to 4A<sub>H</sub> (0 to 74) for a standard diskette and from 00<sub>H</sub> to 21<sub>H</sub> (0 to 33) for a mini-sized diskette.

#### Parameter 1, Sector Address

Specifies the first sector to be scanned. The number of sectors scanned is specified in parameter 2, and the order in which sectors are scanned is specified in parameter 3.

#### Parameter 2, Sector Length/Number of Sectors

The sector length field (bits 7-5) specifies the number of data bytes allocated to each sector (see parameter 2, routine read and write commands for field interpretation). The number of sectors field (bits 4-0) specifies the number of sectors to be scanned. The number specified ranges from one sector to the physical number of sectors on the track.

#### Parameter 3

D<sub>7</sub>-D<sub>6</sub>: Indicate scan type

00-EQ Scan for each character within the field length (key) equal to the corresponding character within the disk sector. The scan stops after the first inequality.

01-GEQ Scan for each character within the disk sector greater than or equal to the corresponding character within the field length (key). The scan stops after the first inequality.

10-LEQ Scan for each character within the disk sector less than or equal to the corresponding character within the field length (key). The scan stops after the first inequality.

D<sub>5</sub>-D<sub>0</sub>: Step Size: The Step Size field specifies the offset to the next sector in a multisector scan. In this case, the next sector address is generated by adding the Step Size to the current sector address.

#### Parameter 4, Field Length

Specifies the number of bytes to be compared (length of key). While the range of legal values is from 1 to 255, the field length specified should be evenly divisible into the sector length to prevent the key from being split at sector boundaries, if the multisector scan commands are used.

#### Scan Command Results

More detailed information about the completion of Scan Commands may be obtained by executing Read Special Register commands.

#### Read Special Register

Parameter (Hex)	Results
06	The <u>sector number</u> of the sector in which the specified scan data pattern was located.
14	MSB Count — The number of 128 byte blocks remaining to be compared in the current sector when the scan data pattern was located. This register is decremented with each 128 byte block read.
13	LSB Count — The number of bytes remaining to be compared in the current sector when the scan data pattern is located. This register is initialized to 128 and is decremented with each byte compared.

#### Commands That Process Special Data

##### Read ID

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	SEL 1	SEL 0	0	1	1	0	1	1
PAR0:	0	1	TRACK ADDRESS							
PAR1:	0	1	0	0	0	0	0	0	0	0
PAR2:	0	1	NUMBER OF ID FIELDS							

The Read ID command transfers the specified number of ID fields into memory (beginning with the first ID field after Index). The CRC character is checked but not transferred.

##### Format Track

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>				
CMD:	0	0	SEL 1	SEL 0	1	0	0	0	1					
PAR0:	0	1	TRACK ADDRESS											
PAR1:	0	1	GAP 3 SIZE MINUS 6											
PAR2:	0	1	SECTOR LENGTH				NO. OF SECTORS/TRACK							
PAR3:	0	1	GAP 5 SIZE MINUS 6											
PAR4:	0	1	GAP 1 SIZE MINUS 6											

The format command can be used to initialize a disk track compatible with the IBM 3740 format. A Shugart "IBM Type" mini-floppy format may also be generated.

The Format command can be used to initialize a diskette, one track at a time. When format command is used, the program must supply ID fields for each sector on the track. During command execution, the supplied ID fields (track head sector addresses and the sector length) are written sequentially on the diskette. The ID address marks originate from the 8271 and is written automatically as the first byte of each ID field. The CRC character is written in the last two bytes of the ID field and is derived from the data written in the first five bytes. During the formatting operation, the data field of each sector is filled with data pattern (E5)<sub>H</sub>. The CRC, derived from the data pattern is also appended to the last byte.

**NOTE:**

1. The parameter 2 (D7-D5) of the Format command specify sector length, the bits are coded the same way as in the Read Data commands.
2. The programmable gap sizes (gap 3, gap 5, and gap 1) must be programmed such that 6 bytes are subtracted from the intended gap size i.e., if gap 1 is intended to be 16 bytes long, programmed length must be  $16 - 6 = 10$  bytes.

The following is the gap size and description summary:

Gap 1 Programmable  
Gap 2 17 Bytes  
Gap 3 Programmable  
Gap 4 Variable  
Gap 5 Programmable

The last six bytes of gaps 1,2,3 and 5 are (00)<sub>H</sub>, all other bytes in the gaps are (FF)<sub>H</sub>. The Gap 1,3 and 5 count specified by the user are the number of bytes of (FF)<sub>H</sub>. Gap 4 is written until the leading edge of the index pulse. If a Gap 5 size of zero is specified, the Index Mark is not written.

## IBM Format Implementation Summary

### Track Format

The disk has 77 tracks, numbered physically from 00 to 76, with track 00 being the outermost track. There are logically 75 data tracks and two alternate tracks. Any two tracks may be initialized as bad tracks. The data tracks are numbered logically in sequence from 00 to 74, skipping over bad tracks.

Note: In IBM format track 00 cannot be a bad track.

### Sector Format

Eack track is divided into 26, 15, or 8 sectors of 128, 256, or 512 bytes length respectively. The first sector is numbered 01, and is physically the first sector after the physical index mark. The logical sequence of the remaining sectors may be nonsequential physically. The location of these is determined at initialization by CPU software.

Each sector consists of an ID field and a data field. All fields are separated by gaps. The beginning of each field is indicated by 6 bytes of (00)<sub>H</sub> followed by a one byte address mark.

### Address Marks

Address Marks are unique bit patterns one byte in length which are used to identify the beginning of ID and Data fields. Address Mark bytes are unique from all other data

bytes in that certain bit cells do not contain a clock bit (all other data bytes have clock bits in every bit cell.) There are four different types of Address Marks used. Each of these is used to identify different types of fields.

### Index Address Mark

The Index Address Mark is located at the beginning of each track and is a fixed number of bytes in front of the first sector.

### ID Address Mark

The ID Address Mark byte is located at the beginning of each ID field on the diskette.

### Data Address Mark

The Data Address Mark byte is located at the beginning of each non-deleted Data Field on the diskette.

### Deleted Data Address Mark

The Deleted Data Address Mark byte is located at the beginning of each deleted Data Field on the diskette.

Address Mark Summary	Clock Pattern	Data Pattern
Index Address Mark	D7	FC
ID Address Mark	C7	FE
Data Address Mark	C7	FB
Deleted Data Address Mark	C7	F8
Bad Track ID Address Mark	C7	FE

### ID Field

MARK	C	H	R	N	CRC	CRC
------	---	---	---	---	-----	-----

C = Cylinder (Track) Address, 00-74

H = Head Address

R = Record (Sector) Address, 01-26

N = Record (Sector) Length, 00-02

Note: Sector Length =  $128 \times 2^N$  bytes

CRC = 16 Bit CRC Character (See Below)

### Data Field

MARK	DATA	CRC	CRC
------	------	-----	-----

Data is 128, 256, or 512 bytes long.

Note: All marks, data, ID characters and CRC characters are recorded and read most significant bit first.

### CRC Character

The 16-bit CRC character is generated using the generator polynomial  $X^{16} + X^{12} + X^5 + 1$ , normally initialized to (FF)<sub>H</sub>. It is generated from all characters (except the CRC in the ID or data field), including the data (not the clocks) in the address mark. It is recorded and read most significant bit first.

### Bad Track Format

The Bad Track Format is the same as the good track format except that the bad track ID field is initialized as follows:

$$C = H = R = N = (FF)_H$$

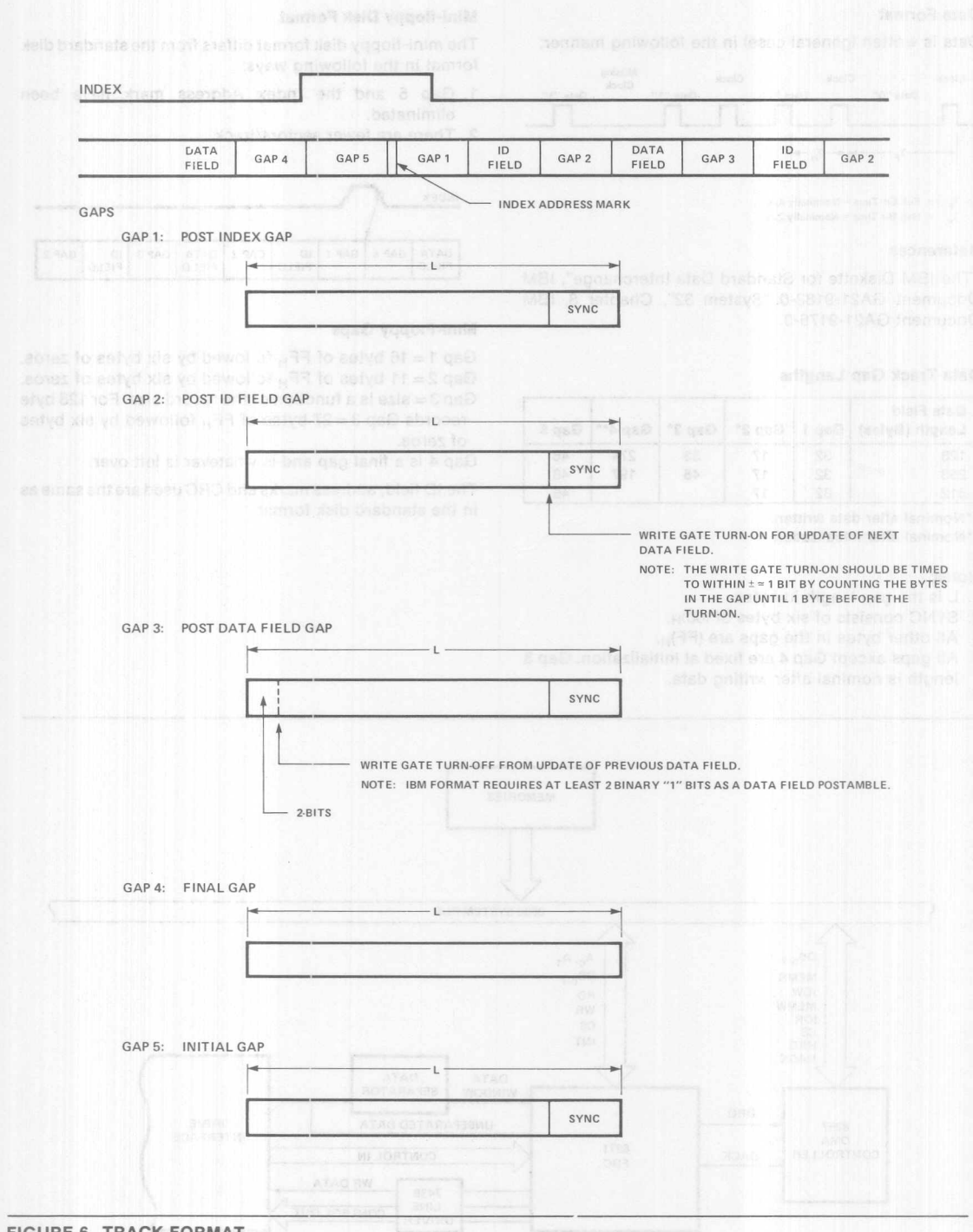
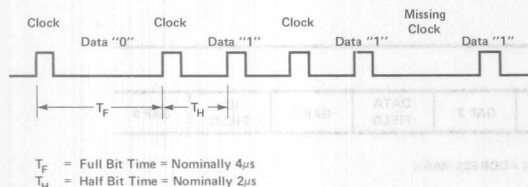


FIGURE 6. TRACK FORMAT

format in the following ways:

1. Gap 5 and the Index Address mark have been eliminated.
2. There are fewer sectors/track.



### References

"The IBM Diskette for Standard Data Interchange", IBM Document GA21-9182-0. "System 32", Chapter 8, IBM Document GA21-9176-0.

### Data Track Gap Lengths

Data Field Length (Bytes)	Gap 1	Gap 2*	Gap 3*	Gap 4**	Gap 5
128	32	17	33	274	46
256	32	17	48	197	46
512	32	17			46

\*Nominal after data written.

\*\*Nominal after initialization.

### Notes:

1. L is the gap length in bytes.
2. SYNC consists of six bytes of (00)<sub>H</sub>.
3. All other bytes in the gaps are (FF)<sub>H</sub>.
4. All gaps except Gap 4 are fixed at initialization. Gap 3 length is nominal after writing data.

### Mini-Floppy Gaps

Gap 1 = 16 bytes of FF<sub>H</sub> followed by six bytes of zeros.  
Gap 2 = 11 bytes of FF<sub>H</sub> followed by six bytes of zeros.  
Gap 3 = size is a function of the record size. For 128 byte records Gap 3 = 27 bytes of FF<sub>H</sub> followed by six bytes of zeros.

Gap 4 is a final gap and is whatever is left over.

The ID field, address marks and CRC used are the same as in the standard disk format.

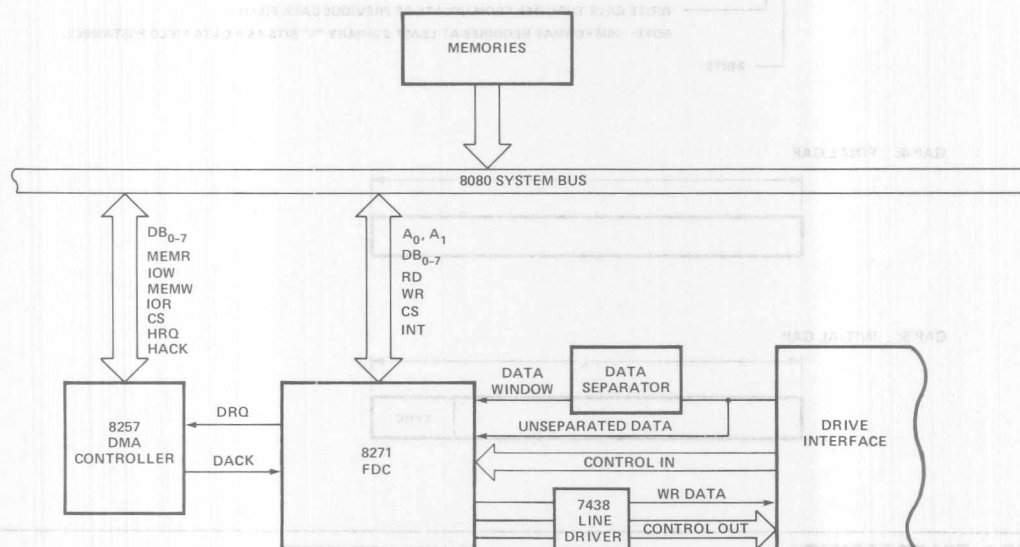


FIGURE 7. 8271 SYSTEM DIAGRAM

**8271 Scan Command Example**

Assume there are only 2 records on track 0 with the following data:

Record 01: 01 02 03 04 05 06 07 08 000....00

Record 02: 01 02 AA 55 00 00 00 00 .....00

Command	Field <sup>[1]</sup> Length	Starting Sector #	# of Sectors	Key <sup>[2]</sup>	Completion Code <sup>[3]</sup>	Special Registers <sup>[4]</sup>			Comment
						R06	R14	R13	
* SCAN EQ	2	1	1	01,02	SME	01	0	127D	Met in first field
SCAN EQ	2	1	1	02,03	SNM	X	X	X	Not met
SCAN EQ	2	1	1	FF <sup>[5]</sup> ,05	SNM	X	X	X	Not met with don't care
* SCAN EQ	2	1	1	FF <sup>[5]</sup> ,06	SME	01	0	123D	Met with don't care
* SCAN EQ	2	1	2	AA,55	SME	02	0	125D	Met in Record 02
* SCAN EQ	2	2	1	01,02	SME	02	0	127D	Starting sector ≠ 1
* SCAN EQ	4	1	1	05,06,07,08	SME	01	0	121D	Field, Key length = 4
* SCAN GEQ	4	1	1	05,06,07,08	SME	01	0	121D	GEQ-SME
* SCAN GEQ	4	1	1	05,04,07,08	SMNE	01	0	121D	GEQ-SMNE
* SCAN GEQ	4	1	2	00,03,AA,44 <sup>[6]</sup>	SNM	X	X	X	GEQ-SNM
* SCAN LEQ	4	1	1	01,03,FF,04	SMNE	01	0	125D	LEQ-SMNE
* SCAN LEQ	4	1	1	01,02,FF,04	SME	01	0	125D	LEQ-SME

**NOTES:**

- Field Length — Each record is partitioned into a number of fields equal to the record size divided by the field length. Note that the record size should be evenly divisible by the field length to insure proper operation of multi record scan. Also, maximum field length = 256 bytes.
- Key — The key is a string of bytes located in the user system memory. The key length should equal the field length. By programming the 8257 DMA Controller into the auto load mode, the key will be recursively read in by the chip (once per field).
- Completion Code — Shows how Scan command was met or not met.  
SNM — SCAN Not Met — 0 0 (also Good Complete)  
SME — SCAN Met Equal — 0 1  
SMNE — SCAN Met Not Equal — 1 0
- Special Registers  
R06 — This register contains the record number where the scan was met.  
R14 — This register contains the MSB count and is decremented every 128 characters.

Length ( $\ell$ ) (D7-D5 of PAR 2)	Record Size	$R14 = 2^{\ell} - 1$ (Initialize at Beginning of Record)
000	128 Bytes	0
001	256 Bytes	1
010	512 Bytes	3
011	1024 Bytes	7
•	•	•
•	•	•
•	•	•

R13 — This register contains a modulo 128 LSB count which is initialized to 128 at beginning of each record. This count is decremented after each character is compared except for the last character in a pattern match situation.

- The FF<sub>H</sub> character in the key is treated as a don't care character position.
- The Scan comparison is done on a byte by byte basis. That is, byte 1 of each field is compared to byte 1 of the key, byte 2 of each field is compared to byte 2 of the key, etc.

**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature Under Bias ..... 0°C to 70°C  
 Storage Temperature ..... -65°C to +150°C  
 Voltage on Any Pin With Respect to Ground ..... -0.5V to +7V  
 Power Dissipation ..... 1 Watt

\*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

**D.C. CHARACTERISTICS**

T<sub>A</sub> = 0°C to 70°C, V<sub>CC</sub> = +5.0V ±5%

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V <sub>IL</sub>	Input Low Voltage	-0.5	0.8	Volts	
V <sub>IH</sub>	Input High Voltage	2.0	(V <sub>CC</sub> + 0.5)	Volts	
V <sub>OL</sub>	Output Low Voltage		0.45	Volts	I <sub>OL</sub> = 2.0mA
V <sub>OH</sub>	Output High Voltage	2.4		Volts	I <sub>OH</sub> = -200μA
I <sub>IL</sub>	Input Load Current		±10	μA	V <sub>IN</sub> = V <sub>CC</sub> to 0V
I <sub>OZ</sub>	Off-State Output Current		±10	μA	V <sub>OUT</sub> = V <sub>CC</sub> to 0V
I <sub>CC</sub>	V <sub>CC</sub> Supply Current		160	mA	

**CAPACITANCE**

T<sub>A</sub> = 25°C; V<sub>CC</sub> = GND = 0V

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
C <sub>IN</sub>	Input Capacitance			10	pF	t <sub>c</sub> = 1MHz
C <sub>I/O</sub>	I/O Capacitance			20	pF	Unmeasured Pins Returned to GND

**PRELIMINARY**  
 Notice: This is not a final specification. Some  
 parametric limits are subject to change.

## A.C. CHARACTERISTICS

$T_A = 0^\circ\text{C to } 70^\circ\text{C}$ ,  $V_{CC} = +5.0\text{V} \pm 5\%$

### Read Cycle

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$t_{AC}$	Select Setup to $\overline{RQ}$	0		ns	
$t_{CA}$	Select Hold from $\overline{RD}$	0		ns	
$t_{RR}$	$\overline{RD}$ Pulse Width	250		ns	
$t_{AD}$	Data Delay from Address		200	ns	
$t_{RD}$	Data Delay from $\overline{RD}$		150	ns	$C_L = 150\text{pF}$
$t_{DF}$	Output Float Delay	20	100	ns	$C_L = 20\text{pF}$ for Minimum; 150pF for Maximum

### Write Cycle

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$t_{AC}$	Select Setup to $\overline{WR}$	0		ns	
$t_{CA}$	Select Hold from $\overline{WR}$	0		ns	
$t_{WW}$	$\overline{WR}$ Pulse Width	250		ns	
$t_{DW}$	Data Setup to $\overline{WR}$	150		ns	
$t_{WD}$	Data Hold from $\overline{WR}$	-20		ns	

### DMA

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$t_{CQ}$	Request Hold from $\overline{WR}$ or $\overline{RD}$ (for Non-Burst Mode)		150	ns	

### Other Timing

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$t_{RSTW}$	Reset Pulse Width	10		tcy	
$t_r$	Input Signal Rise Time		20	ns	
$t_f$	Input Signal Fall Time		20	ns	
$t_{RSTS}$	Reset to First $IOWR$	2		tcy	
$t_{CY}$	Clock Period	250			See Note 3
$t_{CL}$	Clock Low Period				See Note 2
$t_{CH}$	Clock High Period				See Note 2

#### Notes:

1. All timing measurements are made at the following reference voltages unless specified otherwise:

Input "1" at 2.0V, "0" at 0.8V

Output "1" at 2.0V, "0" at 0.8V

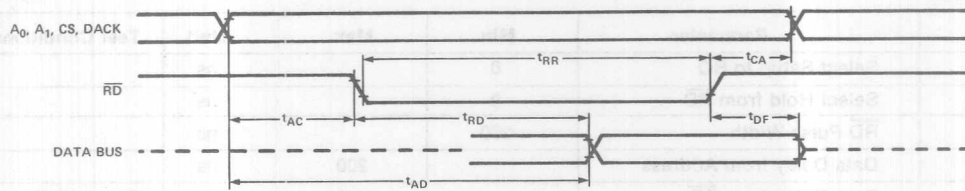
2. To be specified

3. Standard Floppy:  $T_{CY} = 250\text{ns} \pm 0.4\%$

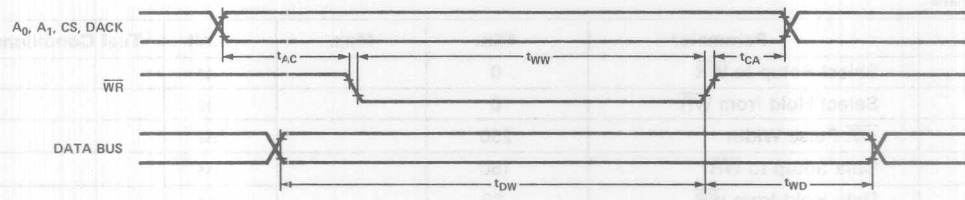
Mini-Floppy:  $T_{CY} = 500\text{ns} \pm 0.4\%$

## WAVEFORMS

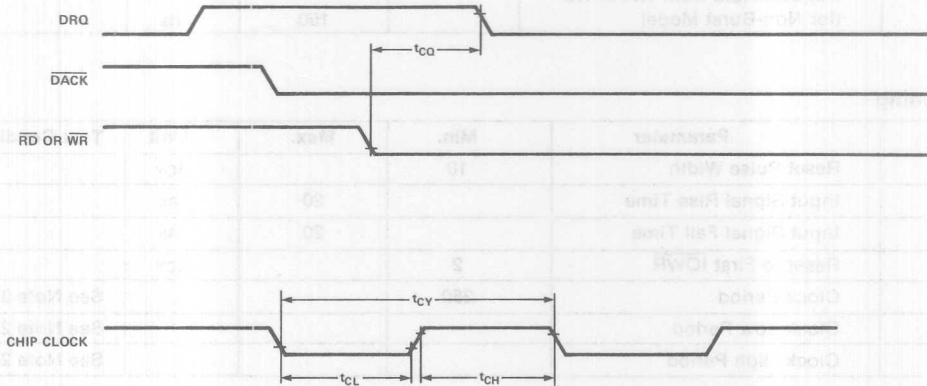
## Read Waveforms



## Write Waveforms



## DMA Waveforms



**PRELIMINARY**  
 Notice: This is not a final specification. Some parametric limits are subject to change.

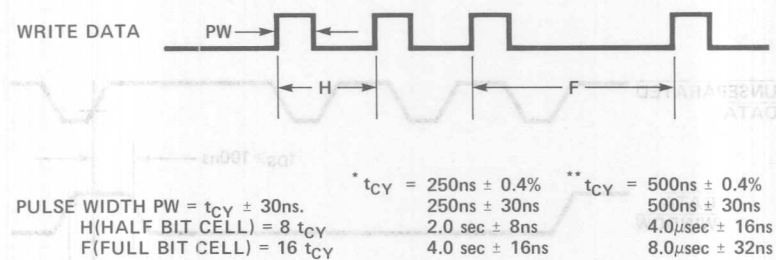


FIGURE 8. WRITE DATA

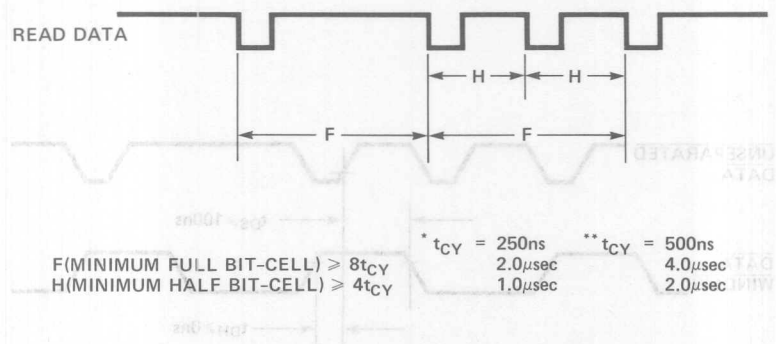


FIGURE 9. READ DATA

- \* Standard flexible disk drive timing.
- \*\* Mini-floppy timing.

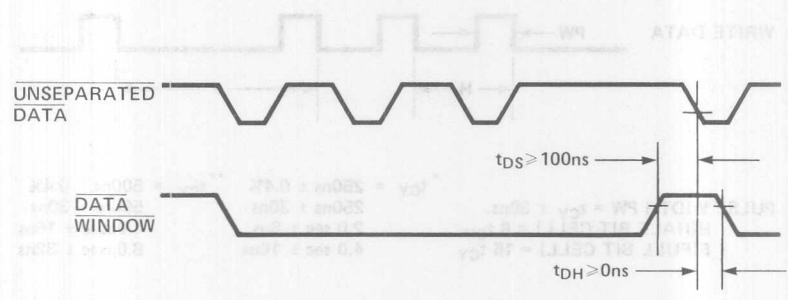


FIGURE 8. WRITE DATA

FIGURE 10. SINGLE-SHOT DATA SEPARATOR

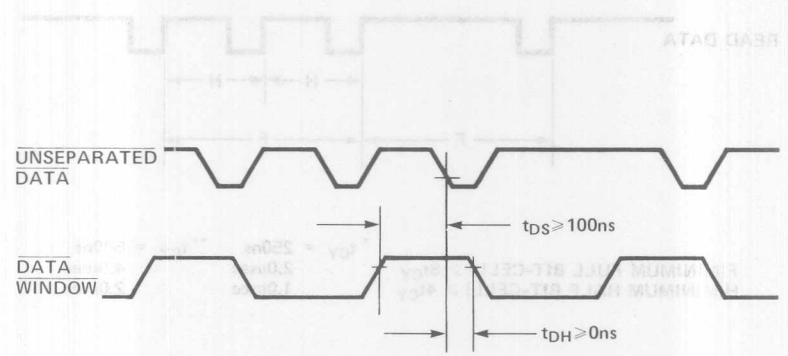


FIGURE 9. READ DATA

FIGURE 11. PLO DATA SEPARATOR

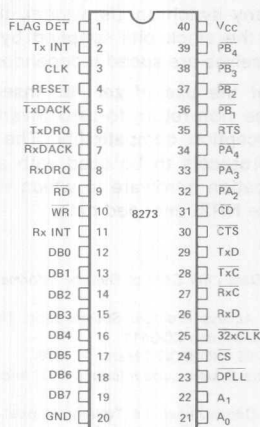
# PROGRAMMABLE HDLC/SDLC PROTOCOL CONTROLLER

**PRELIMINARY**  
Notice: This is not a final specification. Some parametric limits are subject to change.

- HDLC/SDLC Compatible
- Frame Level Commands
- Full Duplex, Half Duplex, or Loop SDLC Operation
- Up to 64K Baud Transfers
- Two User Programmable Modem Control Ports
- Automatic FCS (CRC) Generation and Checking
- Programmable NRZI Encode/Decode
- N-Bit Reception Capability
- Digital Phase Locked Loop Clock Recovery
- Minimum CPU Overhead
- Fully Compatible with 8080/8085 CPUs
- Single +5V Supply
- 40-Pin Package

The Intel® 8273 Programmable HDLC/SDLC Protocol Controller is a dedicated device designed to support the ISO/CITT's HDLC and IBM's SDLC communication line protocols. It is fully compatible with Intel's new high performance microcomputer systems such as the MCS-85™. A frame level command set is achieved by a unique microprogrammed dual processor chip architecture. The processing capability supported by the 8273 relieves the system CPU of the low level real-time tasks normally associated with controllers.

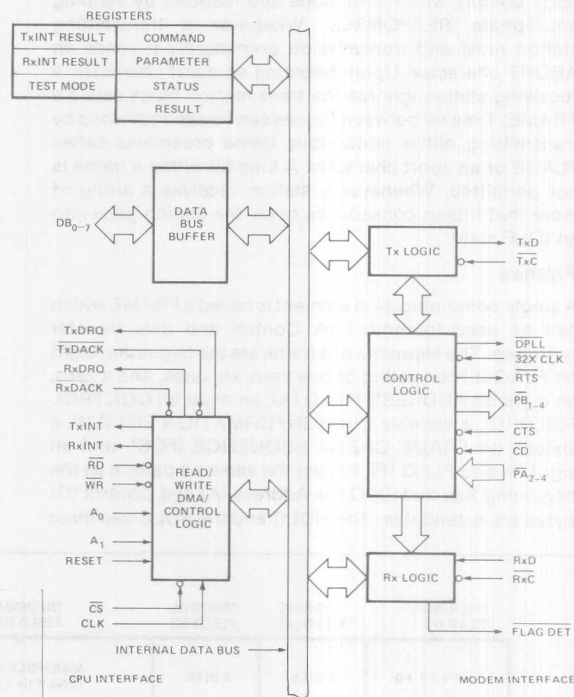
## PIN CONFIGURATION



## PIN NAMES

DB0-DB7	DATA BUS (8 BITS)	CS	CHIP SELECT
FLAG DET	FLAG DETECT	32xCLK	32 TIMES CLOCK
Tx INT	TRANSMITTER INTERRUPT	Rx D	RECEIVER DATA
CLK	CLOCK INPUT	Rx C	RECEIVER CLOCK
RESET	RESET	Tx C	TRANSMITTER CLOCK
Tx DACK	TRANSMITTER DMA ACKNOWLEDGE	Tx D	TRANSMITTER DATA
Tx DRQ	TRANSMITTER DMA REQUEST	CTS	CLEAR TO SEND
RD	READ INPUT	CD	CARRIER DETECT
WR	WRITE INPUT	PA2-PA4	GP INPUT PORTS
Rx DACK	RECEIVER DMA ACKNOWLEDGE	PB1-PB4	GP OUTPUT PORTS
Rx DRQ	RECEIVER DMA REQUEST	RTS	REQUEST TO SEND
Rx INT	RECEIVER INTERRUPT	Vcc	+5 VOLT SUPPLY
A0-A1	COMMAND REGISTER SELECT ADDRESS	GND	GROUND
DPLL	DIGITAL PHASE LOCKED LOOP		

## BLOCK DIAGRAM



## A BRIEF DESCRIPTION OF HDLC/SDLC PROTOCOLS

### General

The High Level Data Link Control (HDLC) is a standard communication link protocol established by International Standards Organization (ISO). HDLC is the discipline used to implement ISO X.25 packet switching systems.

The Synchronous Data Link Control (SDLC) is an IBM communication link protocol used to implement the System Network Architecture (SNA). Both the protocols are bit oriented, code independent, and ideal for full duplex communication. Some common applications include terminal to terminal, terminal to CPU, CPU to CPU, satellite communication, packet switching and other high speed data links. In systems which require expensive cabling and interconnect hardware, any of the two protocols could be used to simplify interfacing (by going serial), thereby reducing interconnect hardware costs. Since both the protocols are speed independent, reducing interconnect hardware could become an important application.

### Network

In both the HDLC and SDLC line protocols, according to a pre-assigned hierarchy, a PRIMARY (Control) STATION controls the overall network (data link) and issues commands to the SECONDARY (Slave) STATIONS. The latter comply with instructions and respond by sending appropriate RESPONSES. Whenever a transmitting station must end transmission prematurely it sends an ABORT character. Upon detecting an abort character, a receiving station ignores the transmission block called a FRAME. Time fill between frames can be accomplished by transmitting either continuous frame preambles called FLAGS or an abort character. A time fill within a frame is not permitted. Whenever a station receives a string of more than fifteen consecutive ones, the station goes into an IDLE state.

### Frames

A single communication element is called a FRAME which can be used for both Link Control and data transfer purposes. The elements of a frame are the beginning eight bit FLAG (F) consisting of one zero, six ones, and a zero, an eight bit ADDRESS FIELD (A), an eight bit CONTROL FIELD (C), a variable (N-bit) INFORMATION FIELD (I), a sixteen bit FRAME CHECK SEQUENCE (FCS), and an eight bit end FLAG (F), having the same bit pattern as the beginning flag. In HDLC the Address (A) and Control (C) bytes are extendable. The HDLC and the SDLC use three

types of frames; an Information Frame is used to transfer data, a Supervisory Frame is used for control purposes, and a Non-sequenced Frame is used for initialization and control of the secondary stations.

### Frame Characteristics

An important characteristic of a frame is that its contents are made code transparent by use of a zero bit insertion and deletion technique. Thus, the user can adopt any format or code suitable for his system — it may even be a computer word length or a "memory dump". The frame is bit oriented that is, bits, not characters in each field, have specific meanings. The Frame Check Sequence (FCS) is an error detection scheme similar to the Cyclic Redundancy Checkword (CRC) widely used in magnetic disk storage devices. The Command and Response information frames contain sequence numbers in the control fields identifying the sent and received frames. The sequence numbers are used in Error Recovery Procedures (ERP) and as implicit acknowledgement of frame communication, enhancing the true full-duplex nature of the HDLC/SDLC protocols.

In contrast, BISYNC is basically half-duplex (two way alternate) because of necessity to transmit immediate acknowledgement frames. HDLC/SDLC therefore saves propagation delay times and have a potential of twice the throughput rate of BISYNC.

It is possible to use HDLC or SDLC over half duplex lines but there is a corresponding loss in throughput because both are primarily designed for full-duplex communication. As in any synchronous system, the bit rate is determined by the clock bits supplied by the modem, protocols themselves are speed independent.

A byproduct of the use of zero-bit insertion-deletion technique is the non-return-to-zero invert (NRZI) data transmission/reception compatibility. The latter allows HDLC/SDLC protocols to be used with asynchronous data communication hardware in which the clocks are derived from the NRZI encoded data.

### References

- IBM Synchronous Data Link Control General Information, IBM, GA27-3093-1.
- Standard Network Access Protocol Specification, DATAPAC, Trans-Canada Telephone System CCG111
- Recommendation X.25, ISO/CCITT March 2, 1976.
- IBM 3650 Retail Store System Loop Interface OEM Information, IBM, GA 27-3098-0
- Guidebook to Data Communications, Training Manual, Hewlett-Packard 5955-1715
- IBM Introduction to Teleprocessing, IBM, GC 20-8095-02
- System Network Architecture, Technical Overview, IBM, GA 27-3102
- System Network Architecture Format and Protocol, IBM GA 27-3112

OPENING FLAG (F)	ADDRESS FIELD (A)	CONTROL FIELD (C)	INFORMATION FIELD (I)	FRAME CHECK SEQUENCE (FCS)	CLOSING FLAG (F)
0 1 1 1 1 1 1 0	8 BITS	3 BITS	VARIABLE LENGTH (ONLY IN I FRAMES)	16 BITS	0 1 1 1 1 1 1 0

Figure 1. Frame Format

## FUNCTIONAL DESCRIPTION

### General

The Intel® 8273 HDLC/SDLC controller is a microcomputer peripheral device which supports the International Standards Organization (ISO) High Level Data Link Control (HDLC), and IBM Synchronous Data Link Control (SDLC) communications protocols. This controller minimizes CPU software by supporting a comprehensive frame-level instruction set and by hardware implementation of the low level tasks associated with frame assembly/disassembly and data integrity. The 8273 can be used in either synchronous or asynchronous applications. In asynchronous applications the data can be programmed to be encoded/decoded in NRZI code. The clock is derived from the NRZI data using a digital phase locked loop. The data transparency is achieved by using a zero-bit insertion/deletion technique. The frames are automatically checked for errors during reception by verifying the Frame Check Sequence (FCS); the FCS is automatically generated and appended before the final flag in transmit. The 8273 recognizes and can generate flags (01111110), Abort, Idle, and GA (EOP) characters.

The 8273 can assume either a primary (control) or a secondary (slave) role. It can therefore be readily implemented in an SDLC loop configuration as typified by the IBM 3650 Retail Store System by programming the 8273 into a one-bit delay mode. In such a configuration, a two wire pair can be effectively used for data transfer between controllers and loop stations. The digital phase locked loop output pin can be used by the loop station without the presence of an accurate Tx clock.

### Hardware Description

The 8273 is packaged in a 40 pin DIP. The following is a functional description of each pin.

Pin Name (No.)	I/O	Description
VCC (40)		+5V Supply
GND (20)		Ground
RESET (4)	I	A high signal on this pin will force the 8273 to an idle state. The 8273 will remain idle until a command is issued by the CPU. The modem interface output signals are forced high. Reset must be true for a minimum of 10 TCY.
$\overline{CS}$ (24)	I	The $\overline{RD}$ and $\overline{WR}$ inputs are enabled by the chip select input.
DB7-DB0 (19-12)	I/O	The Data Bus lines are bidirectional three-state lines which interface with the system Data Bus.
$\overline{WR}$ (10)	I	The Write signal is used to control the transfer of either a command or data from CPU to the 8273.
$\overline{RD}$ (9)	I	The Read signal is used to control the transfer of either a data byte or a status word from the 8273 to the CPU.
TxINT (2)	O	The Transmitter interrupt signal indicates that the transmitter logic requires service.
RxINT (11)	O	The Receiver interrupt signal indicates that the Receiver logic requires service.

TxD $\overline{RQ}$ (6)	O	Requests a transfer of data between memory and the 8273 for a transmit operation.
RxD $\overline{RQ}$ (8)	O	Requests a transfer of data between the 8273 and memory for a receive operation.
$\overline{TxDACK}$ (5)	I	The Transmitter DMA acknowledge signal notifies the 8273 that the TxDMA cycle has been granted.
$\overline{RxDACK}$ (7)	I	The Receiver DMA acknowledge signal notifies the 8273 that the RxDMA cycle has been granted.
A1-A0 (22-21)	I	These two lines are CPU Interface Register Select lines.
TxD (29)	O	This line transmits the serial data to the communication channel.
$\overline{TxC}$ (28)	I	The transmitter clock is used to synchronize the transmit data.
RxD (26)	I	This line receives serial data from the communication channel.
$\overline{RxC}$ (27)	I	The Receiver Clock is used to synchronize the receive data.
32X CLK (25)	I	The 32X clock is used to provide clock recovery when an asynchronous modem is used. In loop configuration the loop station can run without an accurate 1X clock by using the 32X CLK in conjunction with the DPLL output. (This pin must be grounded when not used).
$\overline{DPLL}$ (23)	O	Digital Phase Locked Loop output can be tied to $\overline{RxC}$ and/or $\overline{TxC}$ when 1X clock is not available. DPLL is used with 32X CLK.
$\overline{FLAG DET}$ (1)	O	Flag Detect signals that a flag (01111110) has been received by an active receiver.
$\overline{RTS}$ (35)	O	Request to Send signals that the 8273 is ready to transmit data.
$\overline{CTS}$ (30)	I	Clear to Send signals that the modem is ready to accept data from the 8273.
$\overline{CD}$ (31)	I	Carrier Detect signals that the line transmission has started and the 8273 may begin to sample data on RxD line.
$\overline{PA}_{2-4}$ (32-34)	I	General purpose input ports. The logic levels on these lines can be Read by the CPU through the Data Bus Buffer.
$\overline{PB}_{1-4}$ (36-39)	O	General purpose output ports. The CPU can write these output lines through Data Bus Buffer.
CLK (3)	I	A square wave TTL clock.

The CPU interface is optimized for the MCS-80/85™ bus with an 8257 DMA controller. However, the interface is flexible, and allows either DMA or non-DMA data transfers, interrupt or non-interrupt driven. It further allows maximum line utilization by providing early interrupt mechanism for buffered (only the information field can be transferred to memory) Tx command overlapping. It also provides separate Rx and Tx interrupt output channels for efficient operation. The 8273 keeps the interrupt request active until all the associated interrupt results have been read.

The CPU utilizes the CPU interface to specify commands and transfer data. It consists of seven registers addressed via CS, A<sub>1</sub>, A<sub>0</sub>, RD and WR signals and two independent data registers for receive data and transmit data. A<sub>1</sub>, A<sub>0</sub> are generally derived from two low order bits of the address bus. If an 8080 based CPU is utilized, the RD and WR signals may be driven by the 8228 I/OR and I/OW. The table shows the seven register select decoding:

Address Inputs		Control Logic Inputs	
A <sub>1</sub>	A <sub>0</sub>	CS • RD	CS • WR
0	0	Status	Command
0	1	Result	Parameter
1	0	TxINT Result	Test Mode
1	1	RxINT Result	—

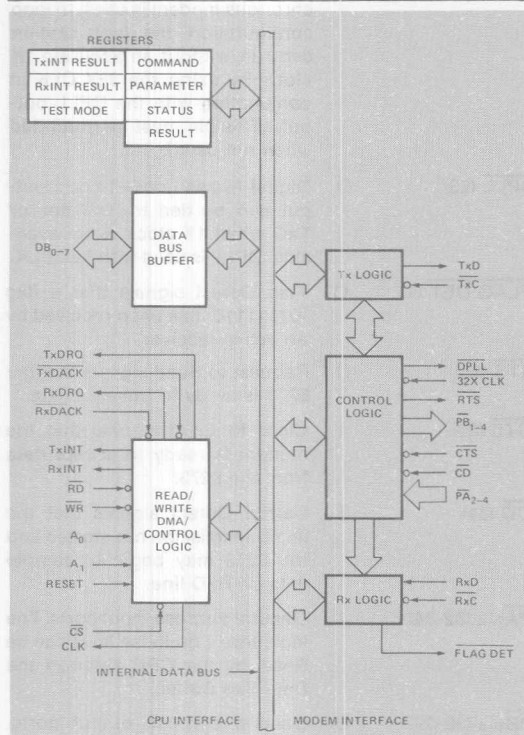


Figure 2. 8273 Block Diagram Showing CPU Interface Functions

## Command

Operations are initiated by writing an appropriate command in the Command Register.

## Parameter

Parameters of commands that require additional information are written to this register.

## Result

Contains an immediate result describing an outcome of an executed command.

## Transmit Interrupt Result

Contains the outcome of 8273 transmit operation (good/bad completion).

## Receive Interrupt Result

Contains the outcome of 8273 receive operation (good/bad completion), followed by additional results which detail the reason for interrupt.

## Status

The status register reflects the state of the 8273 CPU Interface.

## DMA Data Transfers

The 8273 CPU interface supports two independent data interfaces: receive data and transmit data. At high data transmission speeds the data transfer rate of the 8273 is great enough to justify the use of direct memory access (DMA) for the data transfers. When the 8273 is configured in DMA mode, the elements of the DMA interfaces are:

### TxDQ: Transmit DMA Request

Requests a transfer of data between memory and the 8273 for a transmit operation.

### TxDACK: Transmit DMA Acknowledge

The TxDACK signal notifies the 8273 that a transmit DMA cycle has been granted.

### RxDQ: Receive DMA Request

Requests a transfer of data between the 8273 and memory for a receive operation.

**RxDACK: Receive DMA Acknowledge**

The **RxDACK** signal notifies the 8273 that a receive DMA cycle has been granted.

**RD, WR: Read, Write**

The **RD** and **WR** signals are used to specify the direction of the data transfer.

DMA transfers require the use of a DMA controller such as the Intel 8257. The function of the DMA controller is to provide sequential addresses and timing for the transfer, at a starting address determined by the CPU. Counting of data block lengths is performed by the 8273.

To request a DMA transfer the 8273 raises the appropriate DMA REQUEST. DMA ACKNOWLEDGE and READ enables DMA data onto the bus (independently of CHIP SELECT). DMA ACKNOWLEDGE and WRITE transfers DMA data to the 8273 (independent of CHIP SELECT).

It is also possible to configure the 8273 in the non-DMA data transfer mode. In this mode the CPU module must pass data to the 8273 in response to non-DMA data requests indicated by the status word.

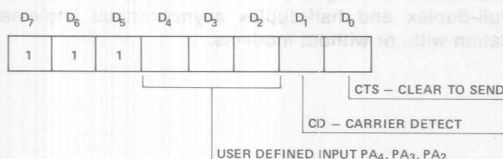
**Modem Interface**

The 8273 Modem interface provides both dedicated and user defined modem control functions. All the signals are active low so that EIA RS-232C inverting drivers (MC 1488) and inverting receivers (MC 1489) may be used to interface to standard modems. For asynchronous operation, this interface supports programmable NRZI data encode/decode, a digital phase locked loop for efficient clock extraction from NRZI data, and modem control ports with automatic **CTS**, **CD** monitoring and **RTS** generation. This interface also allows the 8273 to operate in PRE-FRAME SYNC mode in which the 8273 prefixes 16 transitions to a frame to synchronize idle lines before transmission of the first flag.

It should be noted that all the 8273 port operations deal with logical values, for instance, bit D0 of Port A will be a one when **CTS** (Pin 30) is a physical zero (logical one).

**Port A — Input Port**

During operation, the 8273 interrogates input pins **CTS** (Clear to Send) and **CD** (Carrier Detect). **CTS** is used to condition the start of a transmission. If during transmission **CTS** is lost the 8273 generates an interrupt. During reception, if **CD** is lost, the 8273 generates an interrupt.



The user defined input bits correspond to the 8273 **PA<sub>4</sub>**, **PA<sub>3</sub>** and **PA<sub>2</sub>** pins. The 8273 does not interrogate or manipulate these bits.

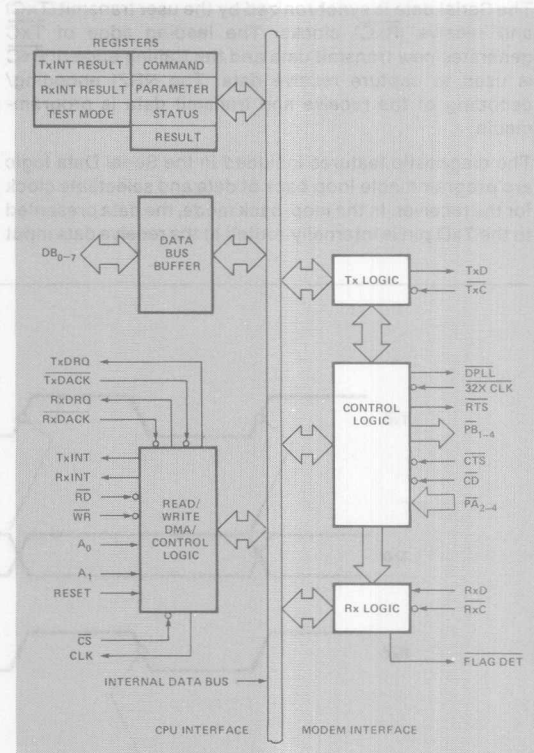
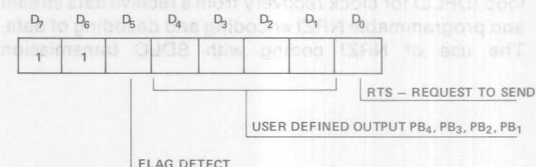


Figure 3. 8273 Block Diagram Showing Control Logic Functions

**Port B - Output Port**

During normal operation, if the CPU sets **RTS** active, the 8273 will not change this pin; however, if the CPU sets **RTS** inactive, the 8273 will activate it before each transmission and deactivate it one byte time after transmission. While the receiver is active the flag detect pin is pulsed each time a flag sequence is detected in the receive data stream. Following an 8273 reset, all pins of Port B are set to a high, inactive level.



The user defined output bits correspond to the state of **PB<sub>4</sub>**-**PB<sub>1</sub>** pins. The 8273 does not interrogate or manipulate these bits.

## Serial Data Logic

The Serial Data is synchronized by the user transmit ( $\overline{\text{TxC}}$ ) and receive ( $\overline{\text{RxC}}$ ) clocks. The leading edge of  $\overline{\text{TxC}}$  generates new transmit data and the trailing edge of  $\overline{\text{RxC}}$  is used to capture receive data. The NRZI encoding/decoding of the receive and transmit data is programmable.

The diagnostic features included in the Serial Data logic are programmable loop back of data and selectable clock for the receiver. In the loop-back mode, the data presented to the  $\overline{\text{TxD}}$  pin is internally routed to the receive data input

circuitry in place of the  $\overline{\text{RxD}}$  pin, thus allowing a CPU to send a message to itself to verify operation of the 8273.

In the selectable clock diagnostic feature, when the data is looped back, the receiver may be presented incorrect sample timing by the external circuitry. The user may select to substitute the  $\overline{\text{TxC}}$  pin for the  $\overline{\text{RxC}}$  input on-chip so that the clock used to generate the loop back data is used to sample it. Since  $\overline{\text{TxD}}$  is generated off the leading edge of  $\overline{\text{TxC}}$  and  $\overline{\text{RxD}}$  is sampled on the trailing edge, the selected clock allows bit synchronism.

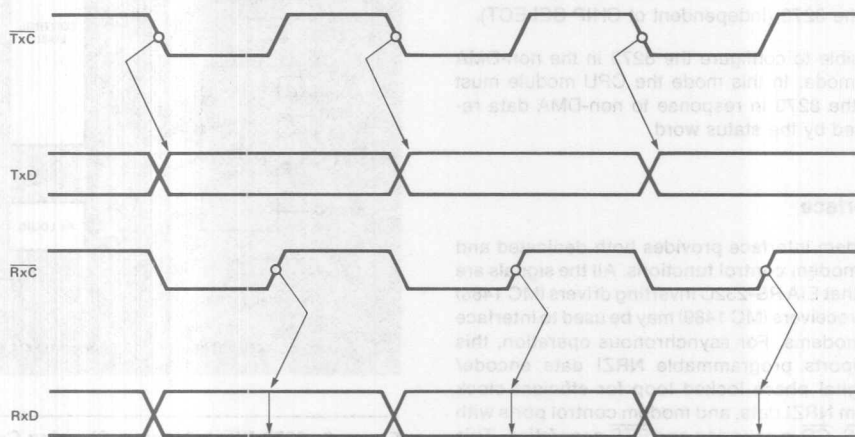


Figure 4. Transmit/Receive Timing

### Asynchronous Mode Interface

Although the 8273 is fully compatible with the HDLC/SDLC communication line protocols, which are primarily designed for synchronous communication, the 8273 can also be used in asynchronous applications by using this interface. The interface employs a digital phase locked loop (DPLL) for clock recovery from a receive data stream and programmable NRZI encoding and decoding of data. The use of NRZI coding with SDLC transmission

guarantees that within a frame, data transitions will occur at least every five bit times — the longest sequence of ones which may be transmitted without zero-bit insertion. The DPLL should be used only when NRZI coding is used since the NRZI coding will transmit zero sequence as line transitions. The digital phase locked loop also facilitates full-duplex and half-duplex asynchronous implementation with, or without modems.

### Digital Phase Locked Loop

In asynchronous applications, the clock is derived from the receiver data stream by the use of the digital phase locked loop (DPLL). The DPLL requires a clock input at 32 times the required baud rate. The receive data (RxD) is sampled with this 32X CLK and the 8273 DPLL supplies a sample pulse nominally centered on the RxD bit cells. The DPLL has a built-in "stiffness" which reduces sensitivity to line noise and bit distortion. This is accomplished by making phase error adjustments in discrete increments. Since the nominal pulse is made to occur at 32 counts of the 32X CLK, these counts are subtracted or added to the nominal, depending upon which quadrant of the four error quadrants the data edge occurs in. For example if an RxD edge is detected in quadrant A1, it is apparent that the DPLL sample "A" was placed too close to the trailing edge of the data cell; sample "B" will then be placed at  $T = (T_{\text{nominal}} - 2 \text{ counts})_i = 30$  counts of the 32X CLK to move the sample pulse "B" toward the nominal center of the next bit cell. A data edge occurring in quadrant B1 would cause a smaller adjustment of phase with  $T = 31$  counts of the 32X CLK. Using this technique the DPLL pulse will converge to nominal bit center within 12 data bit times, worst case, with constant incoming RxD edges.

A method of attaining bit synchronism following a line idle is to use PRE-FRAME SYNC mode of transmission.

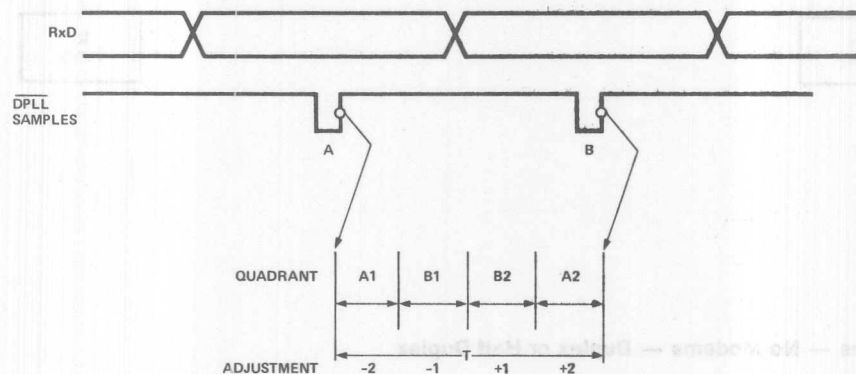
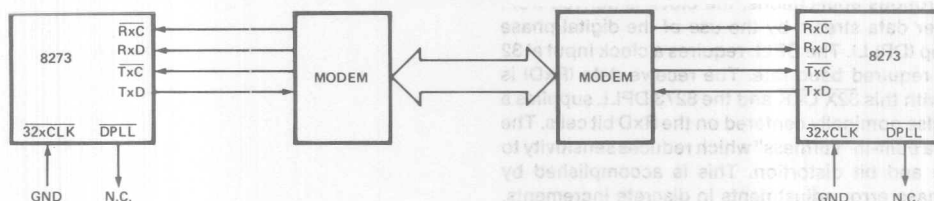
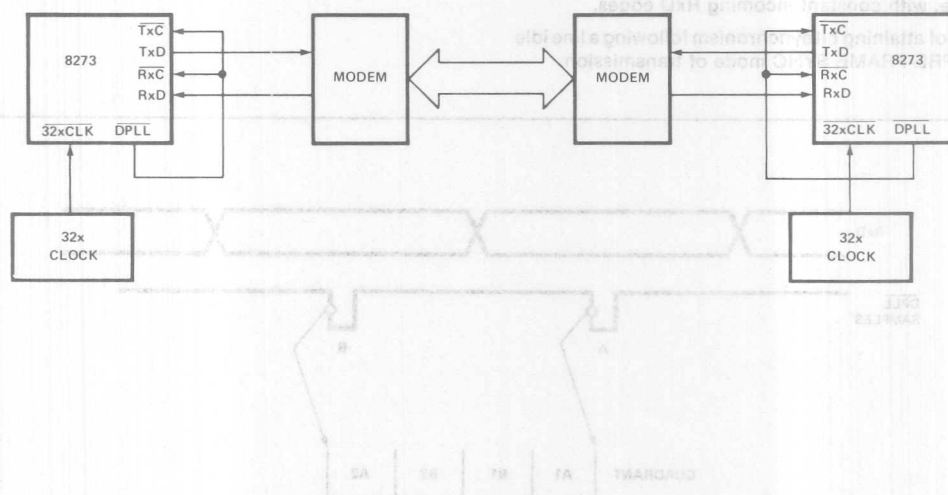


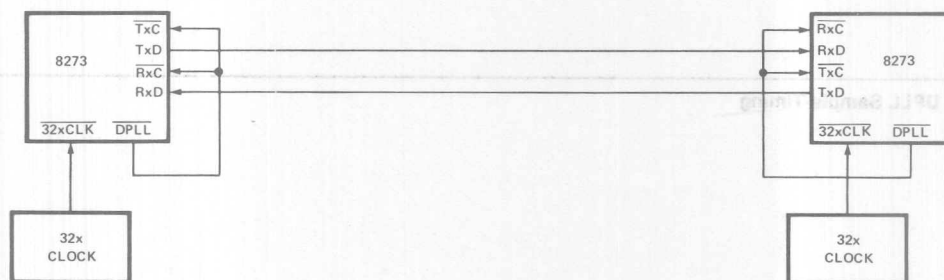
Figure 5. DPLL Sample Timing



### Asynchronous Modems — Duplex or Half Duplex Operation



### Asynchronous — No Modems — Duplex or Half Duplex



### SDLC Loop

The DPLL simplifies the SDLC loop station implementation. In this application, each secondary station on a loop data link is a repeater set in one-bit delay mode. The signals sent out on the loop by the loop controller (primary station) are relayed from station to station then, back to the controller. Any secondary station finding its address in the A field captures the frame for action at that station. All received frames are relayed to the next station on the loop.

Loop stations are required to derive bit timing from the incoming NRZI data stream. The DPLL generates sample Rx clock timing for reception and uses the same clock to implement Tx clock timing.

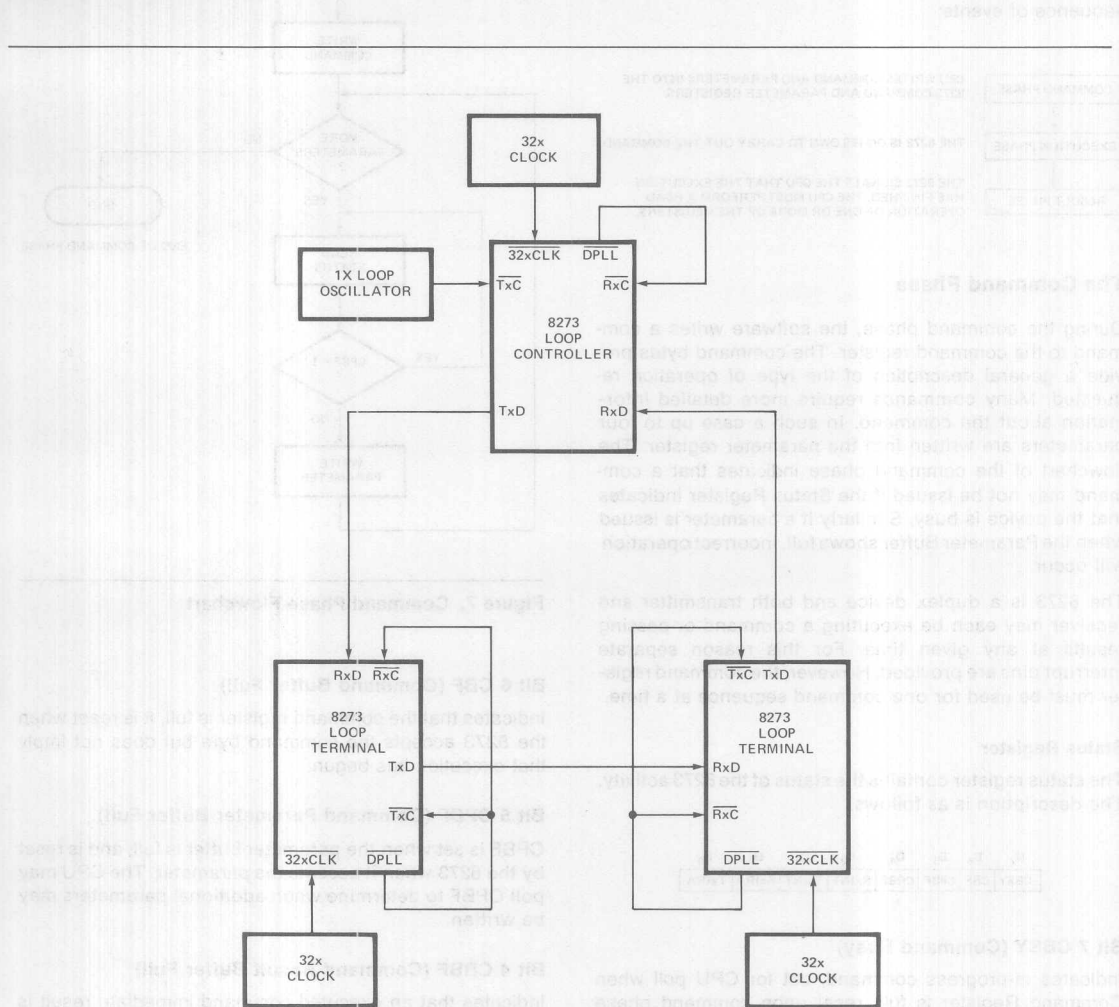
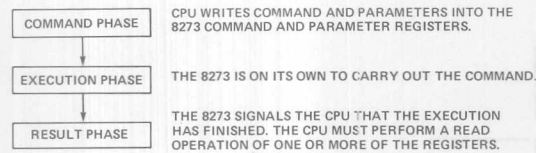


Figure 6. SDLC Loop Application

## PRINCIPLES OF OPERATION

The 8273 is an intelligent peripheral controller which relieves the CPU of many of the rote tasks associated with constructing and receiving frames. It is fully compatible with the MCS-80/85™ system bus. As a peripheral device, it accepts commands from a CPU, executes these commands and provides an Interrupt and Result back to the CPU at the end of the execution. The communication with the CPU is done by activation of  $\overline{CS}$ ,  $\overline{RD}$ ,  $\overline{WR}$  pins, while the  $A_1$ ,  $A_0$  select the appropriate registers on the chip as described in the Hardware Description Section.

The 8273 operation is composed of the following sequence of events:



### The Command Phase

During the command phase, the software writes a command to the command register. The command bytes provide a general description of the type of operation requested. Many commands require more detailed information about the command. In such a case up to four parameters are written into the parameter register. The flowchart of the command phase indicates that a command may not be issued if the Status Register indicates that the device is busy. Similarly if a parameter is issued when the Parameter Buffer shows full, incorrect operation will occur.

The 8273 is a duplex device and both transmitter and receiver may each be executing a command or passing results at any given time. For this reason separate interrupt pins are provided. However, the command register must be used for one command sequence at a time.

### Status Register

The status register contains the status of the 8273 activity. The description is as follows.

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CBSY	CBF	CPBF	CRBF	RxINT	TxINT	RxIRA	TxIRA

#### Bit 7 CBSY (Command Busy)

Indicates in-progress command, set for CPU poll when Command Register is full, reset upon command phase completion. It is improper to write a command when CBSY is set; it results in incorrect operation.

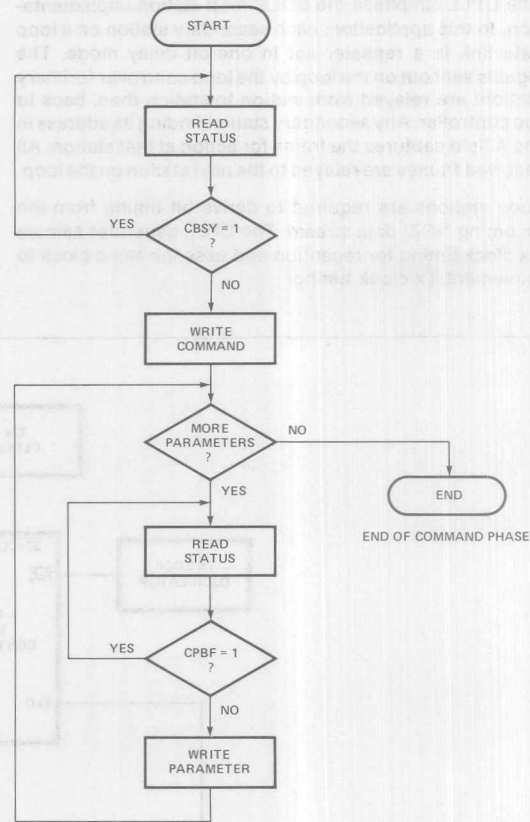


Figure 7. Command Phase Flowchart

#### Bit 6 CBF (Command Buffer Full)

Indicates that the command register is full, it is reset when the 8273 accepts the command byte but does not imply that execution has begun.

#### Bit 5 CPBF (Command Parameter Buffer Full)

CPBF is set when the parameter buffer is full, and is reset by the 8273 when it accepts the parameter. The CPU may poll CPBF to determine when additional parameters may be written.

#### Bit 4 CRBF (Command Result Buffer Full)

Indicates that an executed command immediate result is present in the Result Register. It is set by 8273 and reset when CPU reads the result.

**Bit 3 RxINT (Receiver Interrupt)**

RxINT indicates that the receiver requires CPU attention. It is identical to RxINT (pin 11) and is set by the 8273 either upon good/bad completion of a specified command or by Non-DMA data transfer. It is reset only after the CPU has read the result byte or has received a data byte from the 8273 in a Non-DMA data transfer.

**Bit 2 TxINT (Transmitter Interrupt)**

The TxINT indicates that the transmitter requires CPU attention. It is identical to TxINT (pin 2). It is set by 8273 either upon good/bad completion of a specified command or by Non-DMA data transfer. It is reset only after the CPU has read the result byte or has transferred transmit data byte to the 8273 in a Non-DMA transfer.

**Bit 1 RxIRA (Receiver Interrupt Result Available)**

The RxIRA is set by the 8273 when an interrupt result byte is placed in the RxIRA register. It is reset after the CPU has read the RxIRA register.

**Bit 0 TxIRA (Transmitter Interrupt Result Available)**

The TxIRA is set by the 8273 when an interrupt result byte is placed in the TxIRA register. It is reset when the CPU has read the TxIRA register.

**The Execution Phase**

Upon accepting the last parameter, the 8273 enters into the Execution Phase. The execution phase may consist of a DMA or other activity, and may or may not require CPU intervention. The CPU intervention is eliminated in this phase if the system utilizes DMA for the data transfers, otherwise, for non-DMA data transfers, the CPU is interrupted by the 8273 via TxINT and RxINT pins, for each data byte request.

**The Result Phase**

During the result phase, the 8273 notifies the CPU of the execution outcome of a command. This phase is initiated by:

1. The successful completion of an operation
2. An error detected during an operation.

To facilitate quick network software decisions, two types of execution results are provided:

1. An Immediate Result
2. A Non-Immediate Result

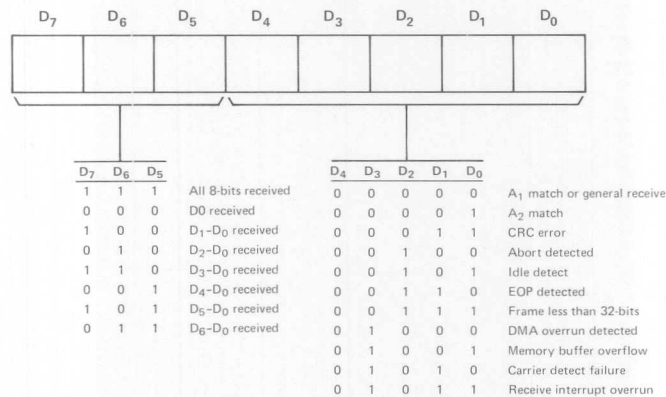


Figure 8. Rx Interrupt Result Byte Format

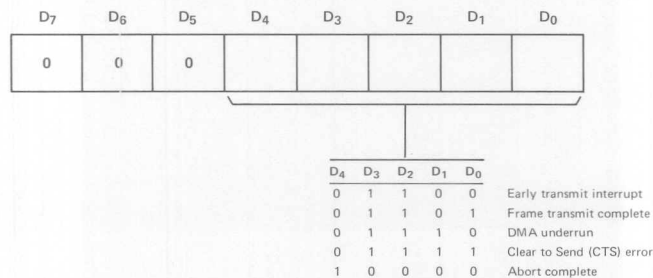


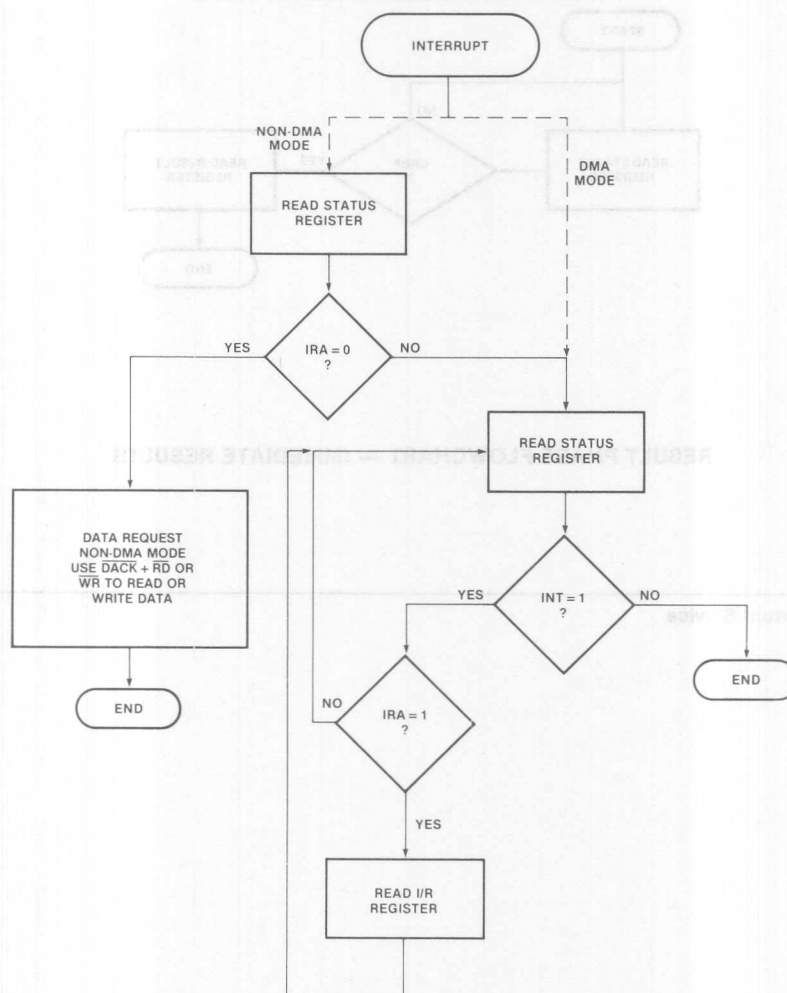
Figure 10. Tx Interrupt Result Byte Format

Immediate result is provided by the 8273 for commands such as Read Port A and Read Port B which have information (CTS, CD, RTS, etc.) that the network software needs to make quick operational decisions.

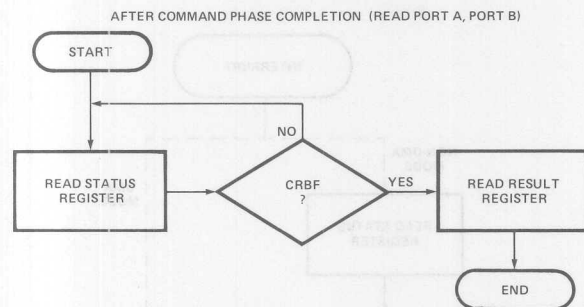
A command which cannot provide an immediate result will generate an interrupt to signal the beginning of the Result phase. The immediate results are provided in the Result Register; all non-immediate results are available upon device interrupt, through Tx Interrupt Result Register TxI/R or Rx Interrupt Result Register RxI/R. The result may consist of a one-byte interrupt code indicating the condition for the interrupt and, if required, one or more bytes which detail the condition.

## Tx and Rx Interrupt Result Registers

The Result Registers have a result code, the three high order bits D7-D5 of which are set to zero for all but the receive command. This command result contains a count that indicates the number of bits received in the last byte. If a partial byte is received, the high order bits of the last data byte are indeterminate.



RESULT PHASE FLOWCHART — INTERRUPT RESULTS



### RESULT PHASE FLOWCHART — IMMEDIATE RESULTS

Figure 9. Rx Interrupt Service

## DETAILED COMMAND DESCRIPTION

### General

The 8273 HDLC/SDLC controller supports a comprehensive set of high level commands which allows the 8273 to be readily used in full-duplex, half-duplex, synchronous, asynchronous and SDLC loop configuration, with or without modems. These frame-level commands minimize CPU and software overhead. The 8273 has address and control byte buffers which allow the receive and transmit commands to be used in buffered or non-buffered modes.

In buffered transmit mode, the 8273 transmits a flag automatically, reads the Address and Control buffer registers and transmits the fields, then via DMA, it fetches the information field. The 8273, having transmitted the information field, automatically appends the Frame Check Sequence (FCS) and the end flag. Correspondingly, in buffered read mode, the Address and Control fields are stored in their respective buffer registers and only Information Field is transferred to memory.

In non-buffered transmit mode, the 8273 transmits the beginning flag automatically, then fetches and transmits the Address, Control and Information fields from the memory, appends the FCS character and an end flag. In the non-buffered receive mode the entire contents of a frame are sent to memory with the exception of the flags and FCS.

### HDLC Implementation

HDLC Address and Control field are extendable. The extension is selected by setting the low order bit of the field to be extended to a one, a zero in the low order bit indicates the last byte of the respective field.

Since Address/Control field extension is normally done with software to maximize extension flexibility, the 8273 does not create or operate upon contents of the extended HDLC Address/Control fields. Extended fields are transparently passed by the 8273 to user as either interrupt results or data transfer requests. Software must assemble the fields for transmission and interrogate them upon reception.

However, the user can take advantage of the powerful 8273 commands to minimize CPU/Software overhead and simplify buffer management in handling extended fields. For instance buffered mode can be used to separate the first two bytes, then interrogate the others from buffer. Buffered mode is perfect for a two byte address field.

The 8273 when programmed, recognizes protocol characters unique to HDLC such as Abort, which is a string of seven or more ones (01111111). Since Abort character is the same as the GA (EOP) character used in SDLC Loop applications, Loop Transmit and Receive commands are not recommended to be used in HDLC. HDLC does not support Loop mode.

### Initialization Set/Reset Commands

These commands are used to manipulate data within the 8273 registers. The Set commands have a single parameter which is a mask that corresponds to the bits to be set. (They perform a logical-OR of the specified register with the mask provided as a parameter). The Register commands have a single parameter which is a mask that has a zero in the bit positions that are to be reset. (They perform a logical-AND of the specified register with the mask).

#### Set One-Bit Delay (CMD Code A4)

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	1	0	1	0	0	1	0	0
PAR:	0	1	1	0	0	0	0	0	0	0

When one bit delay is set, 8273 retransmits the received data stream one bit delayed. This mode is entered at a receiver character boundary, and should only be used by Loop Stations.

#### Reset One-Bit Delay (CMD Code 64)

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	0	1	1	0	0	1	0	0
PAR:	0	1	0	1	1	1	1	1	1	1

The 8273 stops the one bit delayed retransmission mode.

#### Set Data Transfer Mode (CMD Code 97)

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	1	0	0	1	0	1	1	1
PAR:	0	1	0	0	0	0	0	0	0	1

When the data transfer mode is set, the 8273 will interrupt when data bytes are required for transmission or are available from a receive. If a transmit interrupt occurs and the status indicates that there is no Transmit Result (TxIRA = 0), the interrupt is a transmit data request. If a receive interrupt occurs and the status indicates that there is no receive result (RxIRA = 0), the interrupt is a receive data request.

#### Reset Data Transfer Mode (CMD Code 57)

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	0	1	0	1	0	1	1	1
PAR:	0	1	1	1	1	1	1	1	1	0

If the Data Transfer Mode is reset, the 8273 data transfers are performed through the DMA requests without interrupting the CPU.

**Set Operating Mode (CMD Code 91)**

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	1	0	0	1	0	0	0	1
PAR:	0	1	0	0						

1 = FLAG STREAM MODE  
 1 = PREFRAME SYNC MODE  
 1 = BUFFERED MODE  
 1 = EARLY INTERRUPT MODE  
 1 = EOP INTERRUPT MODE  
 1 = HDLC MODE

**Reset Operating Mode (CMD Code 51)**

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	0	1	0	1	0	0	0	1
PAR:	0	1	1	1						

Any mode switches set in CMD code 91 can be reset using this command by placing zeros in the appropriate positions.

**(D5) HDLC Mode**

In HDLC mode, a bit sequence of seven ones (0111111) is interpreted as an abort character. Otherwise, eight ones (01111111) signal an abort.

**(D4) EOP Interrupt Mode**

In EOP interrupt mode, an interrupt is generated whenever an EOP character (01111111) is detected by an active receiver. This mode is useful for the implementation of an SDLC loop controller in detecting the end of a message stream after a loop poll.

**(D3) Transmitter Early Interrupt Mode (Tx)**

The early interrupt mode is specified to indicate when the 8273 should generate an end of frame interrupt. When set, an early interrupt is generated when the last data character has been passed to the 8273. If the user software responds with another transmit command before the final flag is sent, the final flag interrupt will not be generated and a new frame will immediately begin when the current frame is complete. This permits frames to be separated by a single flag. If no additional Tx commands are provided, a final interrupt will follow.

If this bit is zero, the interrupt will be generated only after the final flag has been transmitted.

**(D2) Buffered Mode**

If the buffered mode bit is set to a one, the first two bytes (normally the address (A) and control (C) fields) of a frame are buffered by the 8273. If this bit is a zero the address and control fields are passed to and from memory.

**(D1) Preframe Sync Mode**

If this bit is set to a one the 8273 will transmit two characters before the first flag of a frame. To guarantee sixteen line transitions, the 8273 sends two bytes of data (00)<sub>H</sub> if NRZI is set or data (55)<sub>H</sub> if NRZI is not set.

**(D0) Flag Stream Mode**

If this bit is set to a one, the following table outlines the operation of the transmitter.

TRANSMITTER STATE	ACTION
Idle	Send Flags immediately.
Transmit or Transmit-Transparent Active	Send Flags after the transmission complete.
Loop Transmit Active	Ignore command.
1 Bit Delay Active	Ignore command.

If this bit is reset to zero the following table outlines the operation of the transmitter.

TRANSMITTER STATE	ACTION
IDLE	Send Idles on next character boundary.
Transmit or Transmit-Transparent Active	Send Idles after the transmission is complete.
Loop Transmit Active	Ignore command.
1 Bit Delay Active	Ignore command.

**Set Serial I/O Mode (CMD Code A0)**

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	1	0	1	0	0	0	0	0
PAR:	0	1	0	0	0	0	0			

1 = NRZI MODE  
 1 = Tx C → Rx C  
 1 = LOOP BACK Tx D → Rx D

**Reset Serial I/O Mode (CMD Code 60)**

This command allows bits set in CMD code A0 to be reset by placing zeros in the appropriate positions.

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	0	1	1	1	0	0	0	0
PAR:	0	1	1	1	1	1	1	1		

**(D2) Loop Back**

If this bit is set to a one, the transmit data is internally routed to the receive data circuitry.

**(D1) Tx C → Rx C**

If this bit is set to a one, the transmit clock is internally routed to the receive clock circuitry. It is normally used with the loop back bit (D2).

**(D0) NRZI Mode**

If this bit is set to a one, NRZI encoding and decoding of transmit and receive data is provided. If this bit is a zero, the transmit and receive data is treated as a normal positive logic bit stream.

NRZI encoding specifies that a zero causes a change in the polarity of the transmitted signal and a one causes no polarity change. NRZI is used in all asynchronous operations. Refer to IBM document GA27-3093 for details.

## Reset Device Command

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
TMR:	1	0	0	0	0	0	0	0	0	1
TMR:	1	0	0	0	0	0	0	0	0	0

An 8273 reset command is executed by outputting a (01)<sub>H</sub> followed by (00)<sub>H</sub> to the test mode register (TMR). See 8273 AC timing characteristics for Reset pulse specifications.

The reset command emulates the action of the reset pin.

1. The modem control signals are forced high (inactive level).
2. The 8273 status register flags are cleared.
3. Any commands in progress are terminated immediately.
4. The 8273 enters an idle state until the next command is issued.
5. The Serial I/O and Operating Mode registers are set to zero and DMA data register transfer mode is selected.
6. The device assumes a non-loop SDLC terminal role.

## Receive Commands

The 8273 supports three receive commands: General Receive, Selective Receive, and Selective Loop Receive.

### General Receive (CMD Code C0)

General receive is a receive mode in which frames are received regardless of the contents of the address field.

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>	
CMD:	0	0	1	1	0	0	0	0	0	0	
PAR:	0	1	LEAST SIGNIFICANT BYTE OF THE RECEIVE BUFFER LENGTH (B0)								
PAR:	0	1	MOST SIGNIFICANT BYTE OF RECEIVE BUFFER LENGTH (B1)								

### NOTES:

1. If buffered mode is specified, the R0, R1 receive frame length (result) is the number of data bytes received.
2. If non-buffered mode is specified, the R0, R1 receive frame length (result) is the number of data bytes received plus two (the count includes the address and control bytes).
3. The frame check sequence (FCS) is not transferred to memory.
4. Frames with less than 32 bits between flags are ignored (no interrupt generated) if the buffered mode is specified.
5. In the non-buffered mode an interrupt is generated when a less than 32 bit frame is received, since data transfer requests have occurred.
6. The 8273 receiver is always disabled when an Idle is received after a valid frame. The CPU module must issue a receive command to re-enable the receiver.
7. The intervening ABORT character between a final flag and an IDLE does not generate an interrupt.
8. If an ABORT Character is not preceded by a flag and is followed by an IDLE, an interrupt will be generated for the ABORT followed by an IDLE interrupt one character time later. The reception of an ABORT will disable the receiver.

## Selective Receive (CMD Code C1)

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	1	1	0	0	0	0	0	1
PAR:	0	1	LEAST SIGNIFICANT BYTE OF THE RECEIVE BUFFER LENGTH (B0)							
PAR:	0	1	MOST SIGNIFICANT BYTE OF RECEIVE BUFFER LENGTH (B1)							
PAR:	0	1	RECEIVE FRAME ADDRESS MATCH FIELD ONE (A1)							
PAR:	0	1	RECEIVE FRAME ADDRESS MATCH FIELD TWO (A2)							

Selective receive is a receive mode in which frames are ignored unless the address field matches any one of two address fields given to the 8273 as parameters.

When selective receive is used in HDLC the 8273 looks at the first character, if extended, software must then decide if the message is for this unit.

## Selective Loop Receive (CMD Code C2)

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	1	1	0	0	0	0	1	0
PAR:	0	0	LEAST SIGNIFICANT BYTE OF THE RECEIVE BUFFER LENGTH (B0)							
PAR:	0	1	MOST SIGNIFICANT BYTE OF RECEIVE BUFFER LENGTH (B1)							
PAR:	0	1	RECEIVE FRAME ADDRESS MATCH FIELD ONE (A1)							
PAR:	0	1	RECEIVE FRAME ADDRESS MATCH FIELD TWO (A2)							

Selective loop receive operates like selective receive except that the transmitter is placed in flag stream mode automatically after detecting an EOP (01111111) following a valid received frame. The one bit delay mode is also reset at the end of a selective loop receive.

## Receive Disable (CMD Code C5)

Terminates an active receive command immediately.

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	1	1	0	0	0	1	0	1
PAR:	NONE									

Frame, Loop Transmit, Transmit Transparent.

### Transmit Frame (CMD Code C8)

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	1	1	0	0	1	0	0	0
PAR:	0	1	LEAST SIGNIFICANT BYTE OF FRAME LENGTH (L0)							
PAR:	0	1	MOST SIGNIFICANT BYTE OF FRAME LENGTH (L1)							
PAR:	0	1	ADDRESS FIELD OF TRANSMIT FRAME (A)							
PAR:	0	1	CONTROL FIELD OF TRANSMIT FRAME (C)							

Transmits one frame including: initial flag, frame check sequence, and the final flag.

If the buffered mode is specified, the L0, L1, frame length provided as a parameter is the length of the information field and the address and control fields must be input.

In unbuffered mode the frame length provided must be the length of the information field plus two and the address and control fields must be the first two bytes of data. Thus only the frame length bytes are required as parameters.

### Loop Transmit (CMD Code CA)

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	1	1	0	0	1	0	1	0
PAR:	0	1	LEAST SIGNIFICANT BYTE OF FRAME LENGTH (L0)							
PAR:	0	1	MOST SIGNIFICANT BYTE OF FRAME LENGTH (L1)							
PAR:	0	1	ADDRESS FIELD OF TRANSMIT FRAME (A)							
PAR:	0	1	CONTROL FIELD OF TRANSMIT FRAME (C)							

Transmits one frame in the same manner as the transmit frame command except:

1. This command should be given only in one-bit delay mode.
2. If the flag stream mode is not active transmission will begin after a received EOP has been converted to a flag.
3. If the flag stream mode is active transmission will begin at the next flag boundary for buffered mode or at the third flag boundary for non-buffered mode.
4. At the end of a loop transmit the one-bit delay mode is entered and the flag stream mode is reset.

	CMD:	0	0	1	1	0	0	1	0	0	1
	PAR:	0	1	LEAST SIGNIFICANT BYTE OF FRAME LENGTH (L0)							
	PAR:	0	1	MOST SIGNIFICANT BYTE OF FRAME LENGTH (L1)							

The 8273 will transmit a block of raw data without protocol, i.e., no zero bit insertion, flags, or frame check sequences.

### Abort Transmit Commands

An abort command is supported for each type of transmit command. The abort commands are ignored if a transmit command is not in progress.

#### Abort Transmit Frame (CMD Code CC)

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	1	1	0	0	1	1	0	0
PAR:	NONE									

After an abort character (eight contiguous ones) is transmitted, the transmitter reverts to sending flags or idles as a function of the flag stream mode specified.

#### Abort Loop Transmit (CMD Code CE)

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	1	1	0	0	1	1	1	0
PAR:	NONE									

After a flag is transmitted the transmitter reverts to one bit delay mode.

#### Abort Transmit Transparent (CMD Code CD)

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	1	1	0	0	1	1	0	1
PAR:	NONE									

The transmitter reverts to sending flags or idles as a function of the flag stream mode specified.

## Modem Control Commands

The modem control commands are used to manipulate the modem control ports.

When read Port A or Port B commands are executed the result of the command is returned in the result register. The Bit Set Port B command requires a parameter that is a mask that corresponds to the bits to be set. The Bit Reset Port B command requires a mask that has a zero in the bit positions that are to be reset.

### Read Port A (CMD Code 22)

A<sub>1</sub> A<sub>0</sub> D<sub>7</sub> D<sub>6</sub> D<sub>5</sub> D<sub>4</sub> D<sub>3</sub> D<sub>2</sub> D<sub>1</sub> D<sub>0</sub>  
 CMD: 0 0 0 0 1 0 0 0 1 0  
 PAR: NONE

### Read Port B (CMD Code 23)

A<sub>1</sub> A<sub>0</sub> D<sub>7</sub> D<sub>6</sub> D<sub>5</sub> D<sub>4</sub> D<sub>3</sub> D<sub>2</sub> D<sub>1</sub> D<sub>0</sub>  
 CMD: 0 0 0 0 1 0 0 0 1 1  
 PAR: NONE

### Set Port B Bits (CMD Code A3)

This command allows user defined Port B pins to be set.

A<sub>1</sub> A<sub>0</sub> D<sub>7</sub> D<sub>6</sub> D<sub>5</sub> D<sub>4</sub> D<sub>3</sub> D<sub>2</sub> D<sub>1</sub> D<sub>0</sub>  
 CMD: 0 0 0 1 1 0 0 0 1 1  
 PAR: 0 1 0 0  
 RTS — REQUEST TO SEND  
 USER DEFINED  
 FLAG DETECT

### (D<sub>5</sub>) Flag Detect

This bit can be used to set the flag detect pin. However, it will be reset when the next flag is detected.

### (D<sub>4</sub>-D<sub>1</sub>) User Defined Outputs

These bits correspond to the state of the PB<sub>4</sub>-PB<sub>1</sub> output pins.

### (D<sub>0</sub>) Request to Send

This is a dedicated 8273 modem control signal, and reflects the same logical state of RTS pin.

### Reset Port B Bits (CMD Code 63)

This command allows Port B user defined bits to be reset.

A<sub>1</sub> A<sub>0</sub> D<sub>7</sub> D<sub>6</sub> D<sub>5</sub> D<sub>4</sub> D<sub>3</sub> D<sub>2</sub> D<sub>1</sub> D<sub>0</sub>  
 CMD: 0 0 0 1 1 0 0 0 1 1  
 PAR: 0 1 1 1  
 RTS — REQUEST TO SEND  
 USER DEFINED  
 FLAG DETECT

This command allows Port B (D<sub>4</sub>-D<sub>1</sub>) user defined bits to be reset. These bits correspond to Output Port pins (PB<sub>4</sub>-PB<sub>1</sub>).

## 8273 Command Summary

Command Description	Command (HEX)	Parameter	Results	Result Port	Completion Interrupt
Set One Bit Delay	A4	Set Mask	None	—	No
Reset One Bit Delay	64	Reset Mask	None	—	No
Set Data Transfer Mode	97	Set Mask	None	—	No
Reset Data Transfer Mode	57	Reset Mask	None	—	No
Set Operating Mode	91	Set Mask	None	—	No
Reset Operating Mode	51	Reset Mask	None	—	No
Set Serial I/O Mode	A0	Set Mask	None	—	No
Reset Serial I/O Mode	60	Reset Mask	None	—	No
General Receive	C0	B0,B1	IC,R0,R1,A,C	RXI/R	Yes
Selective Receive	C1	B0,B1,A1,A2	IC,R0,R1,A,C	RXI/R	Yes
Selective Loop Receive	C2	B0,B1,A1,A2	IC,R0,R1,A,C	RXI/R	Yes
Receive Disable	C5	None	None	—	No
Transmit Frame	C8	L0,L1,A,C	IC	TXI/R	Yes
Loop Transmit	CA	L0,L1,A,C	IC	TXI/R	Yes
Transmit Transparent	C9	L0,L1	IC	TXI/R	Yes
Abort Transmit Frame	CC	None	IC	TXI/R	Yes
Abort Loop Transmit	CE	None	IC	TXI/R	Yes
Abort Transmit Transparent	CD	None	IC	TXI/R	Yes
Read Port A	22	None	Port Value	Result	No
Read Port B	23	None	Port Value	Result	No
Set Port B Bit	A3	Set Mask	None	—	No
Reset Port B Bit	63	Reset Mask	None	—	No

## 8273 Command Summary Key

- |              |                                                                                                    |
|--------------|----------------------------------------------------------------------------------------------------|
| <b>B0</b>    | — Least significant byte of the receive buffer length.                                             |
| <b>B1</b>    | — Most significant byte of the receive buffer length.                                              |
| <b>L0</b>    | — Least significant byte of the Tx frame length.                                                   |
| <b>L1</b>    | — Most significant byte of the Tx frame length.                                                    |
| <b>A1</b>    | — Receive frame address match field one.                                                           |
| <b>A2</b>    | — Receive frame address match field two.                                                           |
| <b>A</b>     | — Address field of received frame. If non-buffered mode is specified, this result is not provided. |
| <b>C</b>     | — Control field of received frame. If non-buffered mode is specified this result is not provided.  |
| <b>RXI/R</b> | — Receive interrupt result register.                                                               |
| <b>TXI/R</b> | — Transmit interrupt result register.                                                              |
| <b>R0</b>    | — Least significant byte of the length of the frame received.                                      |
| <b>R1</b>    | — Most significant byte of the length of the frame received.                                       |
| <b>IC</b>    | — Interrupt result code (see table).                                                               |

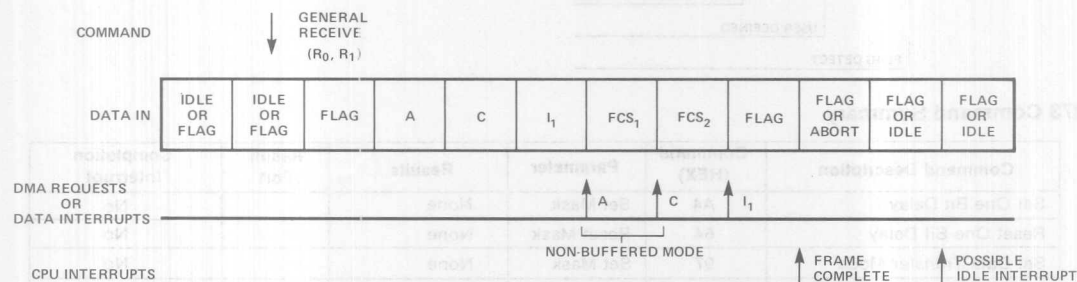


Figure 12. Typical Frame Reception

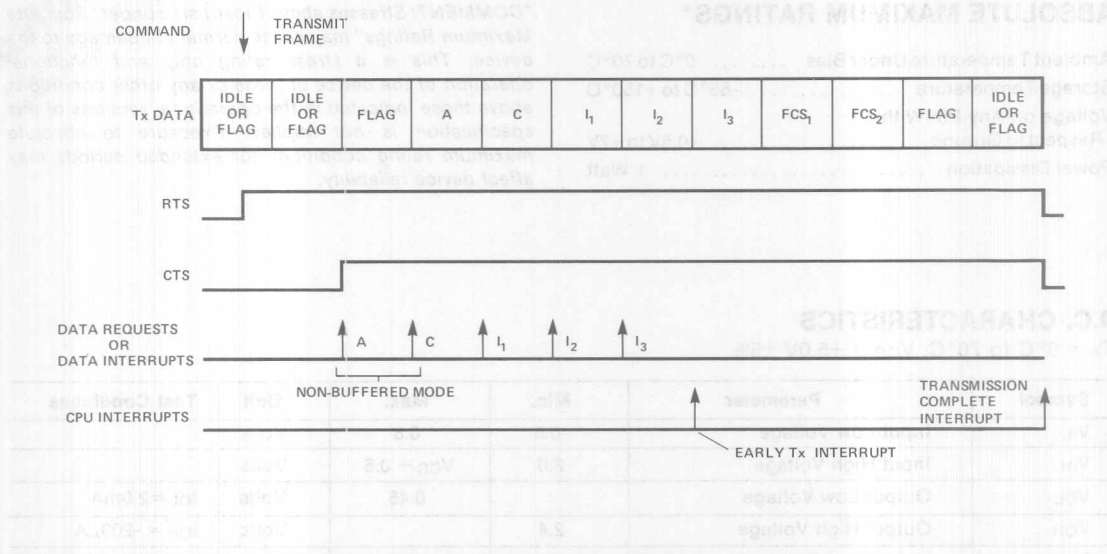


Figure 13. Typical Frame Transmission

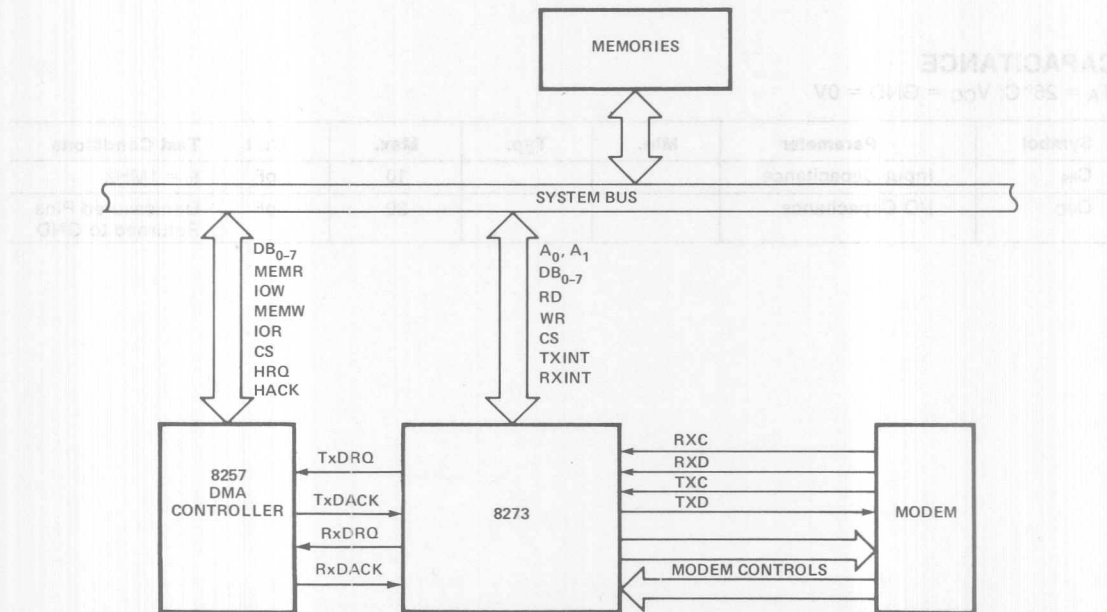


Figure 14. 8273 System Diagram

Ambient Temperature Under Bias ..... 0°C to 70°C  
 Storage Temperature ..... -65°C to +150°C  
 Voltage on Any Pin With  
 Respect to Ground ..... -0.5V to +7V  
 Power Dissipation ..... 1 Watt

Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

## D.C. CHARACTERISTICS

$T_A = 0^\circ\text{C to } 70^\circ\text{C}$ ,  $V_{CC} = +5.0\text{V} \pm 5\%$

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$V_{IL}$	Input Low Voltage	-0.5	0.8	Volts	
$V_{IH}$	Input High Voltage	2.0	$V_{CC} + 0.5$	Volts	
$V_{OL}$	Output Low Voltage		0.45	Volts	$I_{OL} = 2.0\text{mA}$
$V_{OH}$	Output High Voltage	2.4		Volts	$I_{OH} = -200\mu\text{A}$
$I_{IL}$	Input Load Current		$\pm 10$	$\mu\text{A}$	$V_{IN} = V_{CC} \text{ to } 0\text{V}$
$I_{OZ}$	Off-State Output Current		$\pm 10$	$\mu\text{A}$	$V_{OUT} = V_{CC} \text{ to } 0\text{V}$
$I_{CC}$	$V_{CC}$ Supply Current		160	mA	

## CAPACITANCE

$T_A = 25^\circ\text{C}$ ;  $V_{CC} = \text{GND} = 0\text{V}$

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
$C_{IN}$	Input Capacitance			10	pF	$t_c = 1\text{MHz}$
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured Pins Returned to GND

**A.C. CHARACTERISTICS**T<sub>A</sub> = 0°C to 70°C, V<sub>CC</sub> = +5.0V ±5%**Read Cycle**

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t <sub>AC</sub>	Select Setup to RQ	0		ns	
t <sub>CA</sub>	Select Hold from RD	0		ns	
t <sub>RR</sub>	RD Pulse Width	250		ns	
t <sub>AD</sub>	Data Delay from Address		200	ns	
t <sub>RD</sub>	Data Delay from RD		150	ns	C <sub>L</sub> = 150pF
t <sub>DF</sub>	Output Float Delay	20	100	ns	C <sub>L</sub> = 20pF for Minimum; 150pF for Maximum

**Write Cycle**

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t <sub>AC</sub>	Select Setup to WR	0		ns	
t <sub>CA</sub>	Select Hold from WR	0		ns	
t <sub>WW</sub>	WR Pulse Width	250		ns	
t <sub>DW</sub>	Data Setup to WR	150		ns	
t <sub>WD</sub>	Data Hold from WR	-20		ns	

**DMA**

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t <sub>CQ</sub>	Request Hold from WR or RD (for Non-Burst Mode)		150	ns	

**Other Timing**

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t <sub>RSTW</sub>	Reset Pulse Width	10		tcy	
t <sub>r</sub>	Input Signal Rise Time		20	ns	
t <sub>f</sub>	Input Signal Fall Time		20	ns	
t <sub>RSTS</sub>	Reset to First IOWR	2		tcy	
t <sub>CY</sub>	Clock	250			Note 3
t <sub>CL</sub>	Clock Low	T <sub>Bs</sub>			Note 2
t <sub>CH</sub>	Clock High	T <sub>Bs</sub>			Note 2
t <sub>DCL</sub>	Data Clock Low				
t <sub>DCH</sub>	Data Clock High	200		ns	
t <sub>DCY</sub>	Data Clock	15625		ns	Note 3
t <sub>TD</sub>	Transmit Data Delay		100	ns	
t <sub>DS</sub>	Data Setup Time	100		ns	
t <sub>DH</sub>	Data Hold Time	0		ns	
t <sub>DPLL</sub>	DPLL Output Low	200		ns	
t <sub>FLD</sub>	FLAG DET Output Low	8·t <sub>cy</sub> ±50		ns	

**NOTES:**

1. All timing measurements are made at the reference voltages unless otherwise specified:

Input "1" at 2.0V, "0" at 0.8V

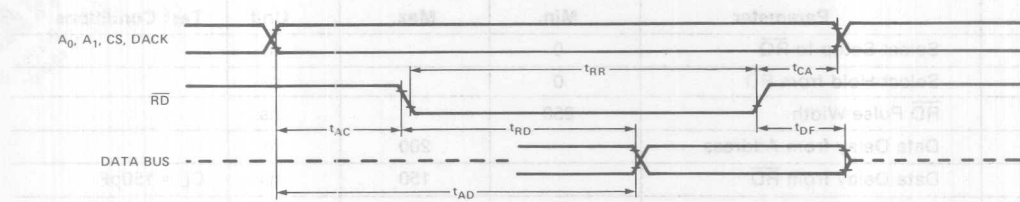
Output "1" at 2.0V, "0" at 0.8V

2. To be specified.

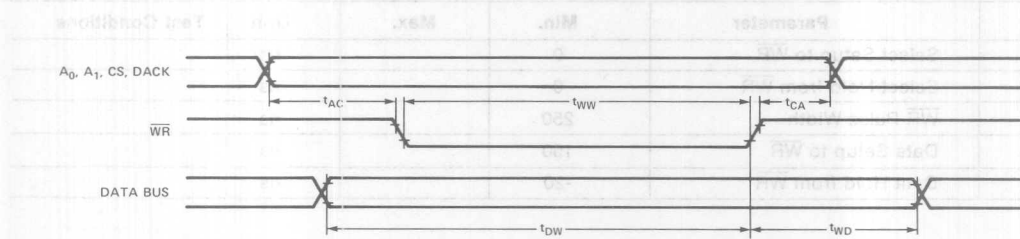
3. 64K baud maximum operating rate.

## WAVEFORMS

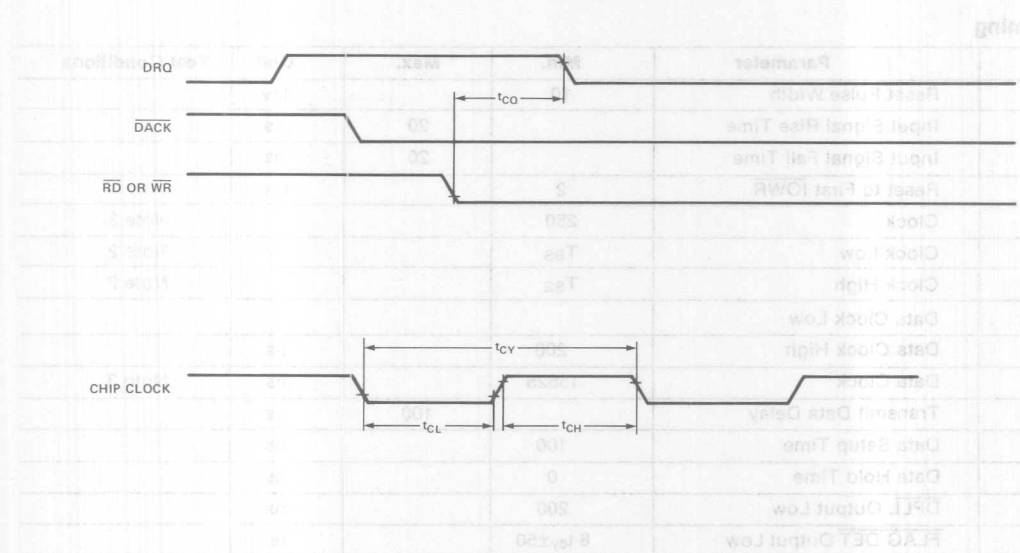
## Read Waveforms



## Write Waveforms



## DMA Waveforms



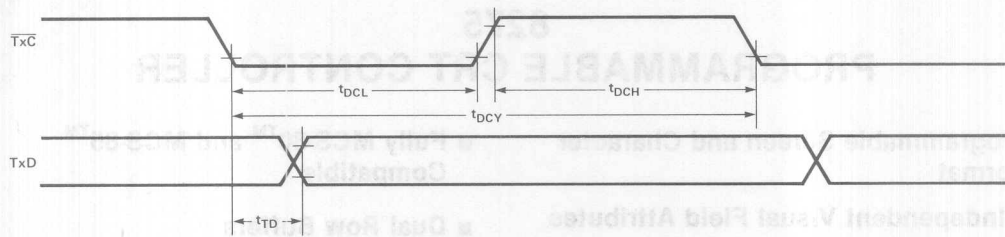
NOTES:  
1. All timing measurements are made at the reference voltages unless otherwise specified.

Input "1" at 2.0V, "0" at 0.8V.  
Output "1" at 2.0V, "0" at 0.8V.

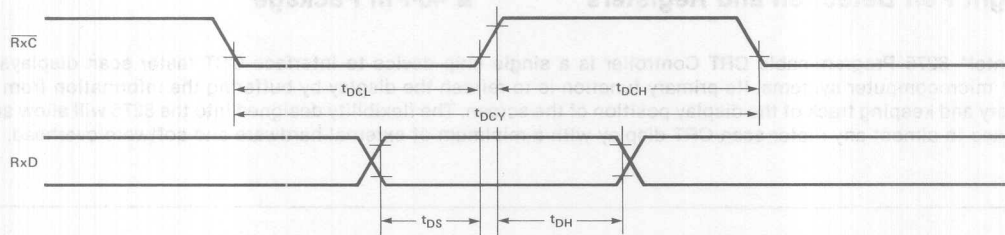
2. To be specified.

3. Data rate maximum operating rate.

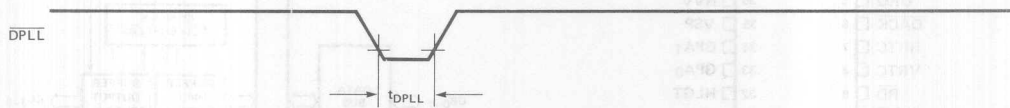
## Transmit Data Waveforms



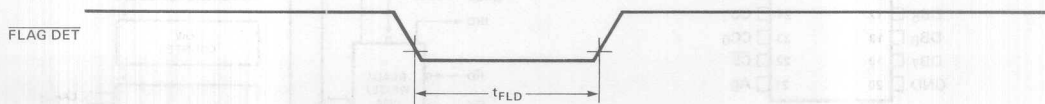
## Receive Data Waveforms



## DPLL Output Waveform



## Flag Detect Output Waveform

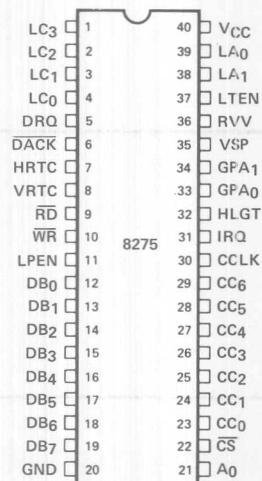


# 8275 PROGRAMMABLE CRT CONTROLLER

- Programmable Screen and Character Format
- 6 Independent Visual Field Attributes
- 11 Visual Character Attributes (Graphic Capability)
- Cursor Control (4 Types)
- Light Pen Detection and Registers
- Fully MCS-80™ and MCS-85™ Compatible
- Dual Row Buffers
- Programmable DMA Burst Mode
- Single +5V Supply
- 40-Pin Package

The Intel® 8275 Programmable CRT Controller is a single chip device to interface CRT raster scan displays with Intel® microcomputer systems. Its primary function is to refresh the display by buffering the information from main memory and keeping track of the display position of the screen. The flexibility designed into the 8275 will allow simple interface to almost any raster scan CRT display with a minimum of external hardware and software overhead.

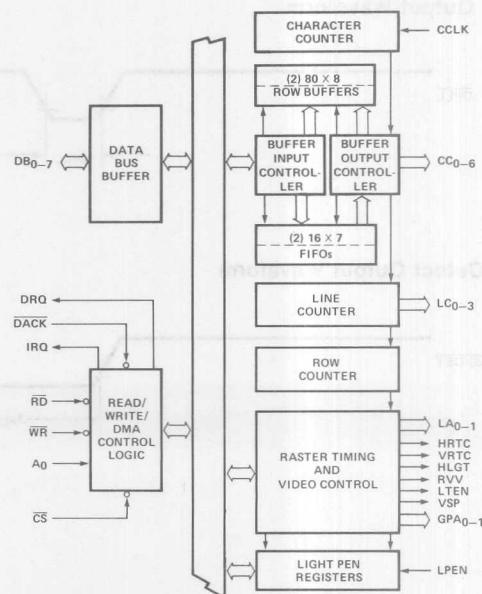
PIN CONFIGURATION



PIN NAMES

DB0-1	B1-DIRECTIONAL DATA BUS	LC0-3	LINE COUNTER OUTPUTS
DRQ	DMA REQUEST OUTPUT	LA0-1	LINE ATTRIBUTE OUTPUTS
DACK	DMA ACKNOWLEDGE INPUT	HRTC	HORIZONTAL RETRACE OUTPUT
IRQ	INTERRUPT REQUEST OUTPUT	VRTC	VERTICAL RETRACE OUTPUT
RD	READ STROBE INPUT	HLGT	HIGHLIGHT OUTPUT
WR	WRITE STROBE INPUT	RVV	REVERSE VIDEO OUTPUT
A0	REGISTER ADDRESS INPUT	LTEN	LIGHT ENABLE OUTPUT
CS	CHIP SELECT INPUT	VSP	VIDEO SUPPRESS OUTPUT
CCLK	CHARACTER CLOCK INPUT	GPA0-1	GENERAL PURPOSE ATTRIBUTE OUTPUTS
CC0-6	CHARACTER CODE OUTPUTS	LPEN	LIGHT PEN INPUT

BLOCK DIAGRAM



## PIN DESCRIPTIONS

Pin #	Pin Name	I/O	Pin Description	Pin #	Pin Name	I/O	Pin Description
1	LC <sub>3</sub>	O	Line count. Output from the line counter which is used to address the character generator for the line positions on the screen.	40	V <sub>CC</sub>		+5V power supply
2	LC <sub>2</sub>			39	LA <sub>0</sub>	O	Line attribute codes. These attribute codes have to be decoded externally by the dot/timing logic to generate the horizontal and vertical line combinations for the graphic displays specified by the character attribute codes.
3	LC <sub>1</sub>			38	LA <sub>1</sub>		
4	LC <sub>0</sub>			37	LTEN	O	Light enable. Output signal used to enable the video signal to the CRT. This output is active at the programmed underline cursor position, and at positions specified by attribute codes.
5	DRQ	O	DMA request. Output signal to the 8257 DMA controller requesting a DMA cycle.	36	RVV	O	Reverse video. Output signal used to indicate the CRT circuitry to reverse the video signal. This output is active at the cursor position if a reverse video block cursor is programmed or at the positions specified by the field attribute codes.
6	DACK	I	DMA acknowledge. Input signal from the 8257 DMA controller acknowledging that the requested DMA cycle has been granted.	35	VSP	O	Video suppression. Output signal used to blank the video signal to the CRT. This output is active: <ul style="list-style-type: none"> <li>— during the horizontal and vertical retrace intervals.</li> <li>— at the top and bottom lines of rows if underline is programmed to be number 8 or greater.</li> <li>— when an end of row or end of screen code is detected.</li> <li>— When a DMA underrun occurs.</li> <li>— at regular intervals (1/16 frame frequency for cursor, 1/32 frame frequency for character and field attributes) — to create blinking displays as specified by cursor, character attribute, or field attribute programming.</li> </ul>
7	HRTC	O	Horizontal retrace. Output signal which is active during the programmed horizontal retrace interval. During this period the VSP output is high and the LTEN output is low.	34	GPA <sub>1</sub>	O	General purpose attribute codes. Outputs which are enabled by the general purpose field attribute codes.
8	VRTC	O	Vertical retrace. Output signal which is active during the programmed vertical retrace interval. During this period the VSP output is high and the LTEN output is low.	33	GPA <sub>0</sub>		
9	RD	I	Read input. A control signal to read registers.	32	HLGT	O	Highlight. Output signal used to intensify the display at particular positions on the screen as specified by the character attribute codes or field attribute codes.
10	WR	I	Write input. A control signal to write commands into the control registers or write data into the row buffers during a DMA cycle.	31	IRQ	O	Interrupt request.
11	LPEN	I	Light pen. Input signal from the CRT system signifying that a light pen signal has been detected.	30	CCLK	I	Character clock (from dot/timing logic).
12	DB <sub>0</sub>	I/O	Bi-directional three-state data bus lines. The outputs are enabled during a read of the C or P ports.	29	CC <sub>6</sub>	O	Character codes. Output from the row buffers used for character selection in the character generator.
13	DB <sub>1</sub>			28	CC <sub>5</sub>		
14	DB <sub>2</sub>			27	CC <sub>4</sub>		
15	DB <sub>3</sub>			26	CC <sub>3</sub>		
16	DB <sub>4</sub>			25	CC <sub>2</sub>		
17	DB <sub>5</sub>			24	CC <sub>1</sub>		
18	DB <sub>6</sub>			23	CC <sub>0</sub>		
19	DB <sub>7</sub>			22	CS	I	Chip select. The read and write are enabled by CS.
20	Ground		Ground	21	A <sub>0</sub>	I	Port address. A high input on A <sub>0</sub> selects the "C" port or command registers and a low input selects the "P" port or parameter registers.

## FUNCTIONAL DESCRIPTION

### Data Bus Buffer

This 3-state, bidirectional, 8-bit buffer is used to interface the 8275 to the system Data Bus.

This functional block accepts inputs from the System Control Bus and generates control signals for overall device operation. It contains the Command, Parameter, and Status Registers that store the various control formats for the device functional definition.

A <sub>0</sub>	OPERATION	REGISTER
0	Read	PREG
0	Write	PREG
1	Read	SREG
1	Write	CREG

### RD (Read)

A "low" on this input informs the 8275 that the CPU is reading data or status information from the 8275.

### WR (Write)

A "low" on this input informs the 8275 that the CPU is writing data or control words to the 8275.

### CS (Chip Select)

A "low" on this input selects the 8275. No reading or writing will occur unless the device is selected. When  $\overline{CS}$  is high, the Data Bus in the float state and RD and WR will have no effect on the chip.

### DRQ (DMA Request)

A "high" on this output informs the DMA Controller that the 8275 desires a DMA transfer.

### DACK (DMA Acknowledge)

A "low" on this input informs the 8275 that a DMA cycle is in progress.

### IRQ (Interrupt Request)

A "high" on this output informs the CPU that the 8275 desires interrupt service.

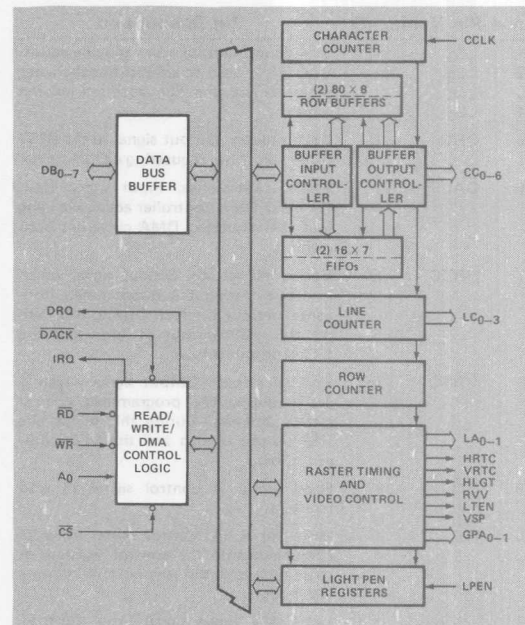


Figure 1. 8275 Block Diagram Showing Data Bus Buffer and Read/Write Functions

A <sub>0</sub>	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	
0	0	1	0	Write 8275 Parameter
0	1	0	0	Read 8275 Parameter
1	0	1	0	Write 8275 Command
1	1	0	0	Read 8275 Status
X	1	1	0	Three-State
X	X	X	1	Three-state

### Character Counter

The Character Counter is a programmable counter that is used to determine the number of characters to be displayed per row and the length of the horizontal retrace interval. It is driven by the CCLK (Character Clock) input, which should be a derivative of the external dot clock.

### Line Counter

The Line Counter is a programmable counter that is used to determine the number of horizontal lines (Sweeps) per character row. Its outputs are used to address the external character generator ROM.

### Row Counter

The Row Counter is a programmable counter that is used to determine the number of character rows to be displayed per frame and length of the vertical retrace interval.

### Light Pen Registers

The Light Pen Registers are two registers that store the contents of the character counter and the row counter whenever there is a rising edge on the LPEN (Light Pen) input.

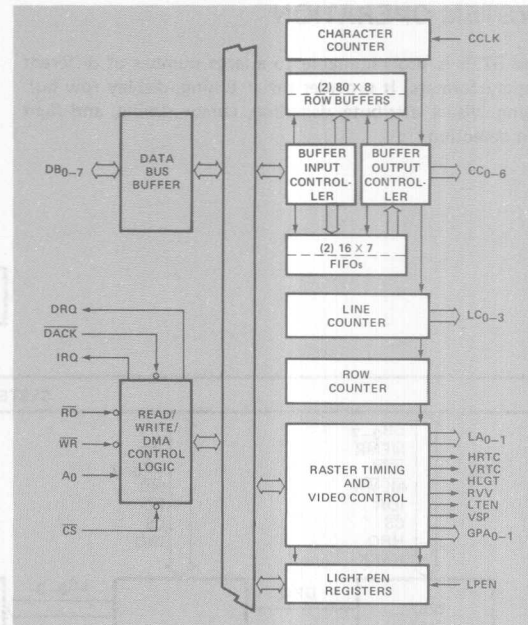
**Note:** Software correction is required.

### Raster Timing and Video Controls

The Raster Timing circuitry controls the timing of the HRTC (Horizontal Retrace) and VRTC (Vertical Retrace) outputs. The Video Control circuitry controls the generation of LA<sub>0-1</sub> (Line Attribute), HGLT (Highlight), RVV (Reverse Video), LTEN (Light Enable), VSP (Video Suppress), and GPA<sub>0-1</sub> (General Purpose Attribute) outputs.

### Row Buffers

The Row Buffers are two 80 character buffers. They are filled from the microcomputer system memory with the character codes to be displayed. While one row buffer is displaying a row of characters, the other is being filled with the next row of characters.



**Figure 2. 8275 Block Diagram Showing Counter and Register Functions**

### FIFOs

There are two 16 character FIFOs in the 8275. They are used to provide extra row buffer length in the Transparent Attribute Mode (see Detailed Operation section).

### Buffer Input/Output Controllers

The Buffer Input/Output Controllers decode the characters being placed in the row buffers. If the character is a character attribute, field attribute or special code, these controllers control the appropriate action. (Examples: An "End of Screen—Stop DMA" special code will cause the Buffer Input Controller to stop further DMA requests. A "Highlight" field attribute will cause the Buffer Output Controller to activate the HGLT output.)

The 8275 is programmable to a large number of different display formats. It provides raster timing, display row buffering, visual attribute decoding, cursor timing, and light pen detection.

It is designed to interface with the 8257 DMA Controller and standard character generator ROMs for dot matrix decoding. Dot level timing must be provided by external circuitry.

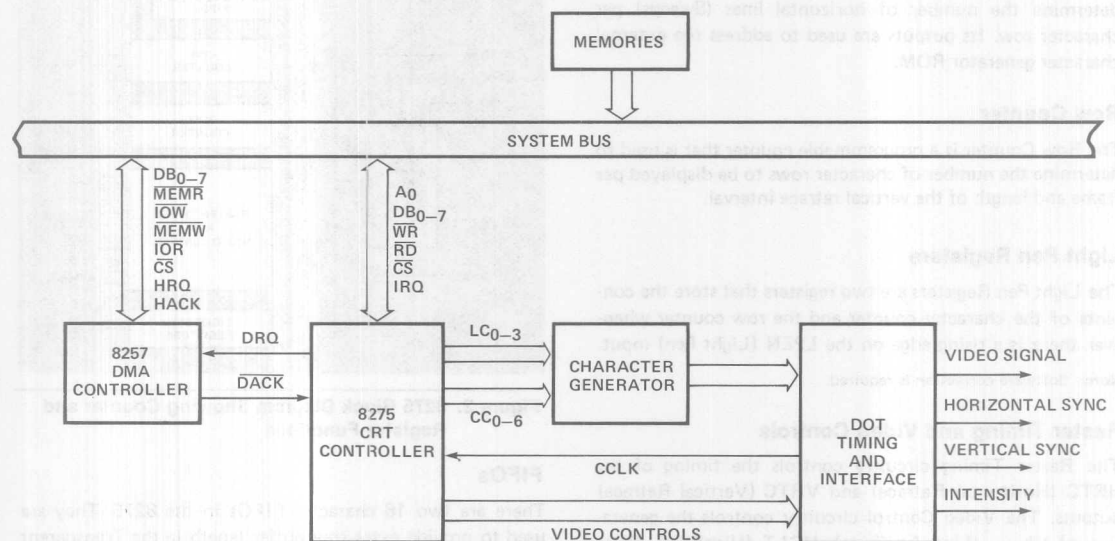


Figure 3. 8275 Systems Block Diagram Showing Systems Operation

## General Systems Operational Description

The 8275 provides a "window" into the microcomputer system memory.

Display characters are retrieved from memory and displayed on a row by row basis. The 8275 has two row buffers. While one row buffer is being used for display, the other is being filled with the next row of characters to be displayed. The number of display characters per row and the number of character rows per frame are software programmable, providing easy interface to most CRT displays. (See Programming Section.)

The 8275 requests DMA to fill the row buffer that is not being used for display. DMA burst length and spacing is programmable. (See Programming Section.)

The 8275 displays character rows one line at a time.

The number of lines per character row, the underline position, and blanking of top and bottom lines are programmable. (See Programming Section.)

The 8275 provides special Control Codes which can be used to minimize DMA or software overhead. It also provides Visual Attribute Codes to cause special action or symbols on the screen without the use of the character generator (see Visual Attributes Section).

The 8275 also controls raster timing. This is done by generating Horizontal Retrace (HRTC) and Vertical Retrace (VRTC) signals. The timing of these signals is programmable.

The 8275 can generate a cursor. Cursor location and format are programmable. (See Programming Section.)

The 8275 has a light pen input and registers. The light pen input is used to load the registers. Light pen registers can be read on command. (See Programming Section.)

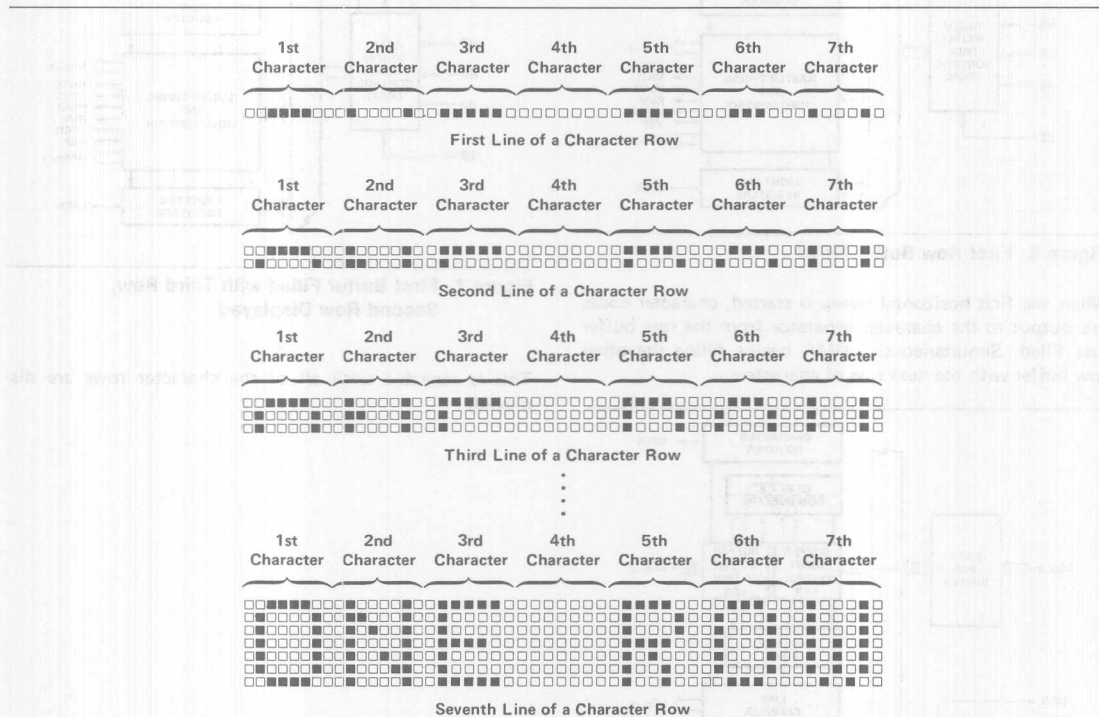


Figure 4. Display of a Character Row

### Display Row Buffering

Before the start of a frame, the 8275 requests DMA and one row buffer is filled with characters.

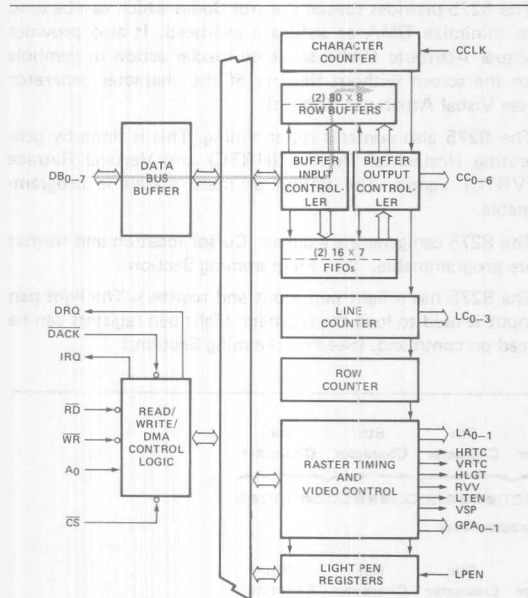


Figure 5. First Row Buffer Filled

When the first horizontal sweep is started, character codes are output to the character generator from the row buffer just filled. Simultaneously, DMA begins filling the other row buffer with the next row of characters.

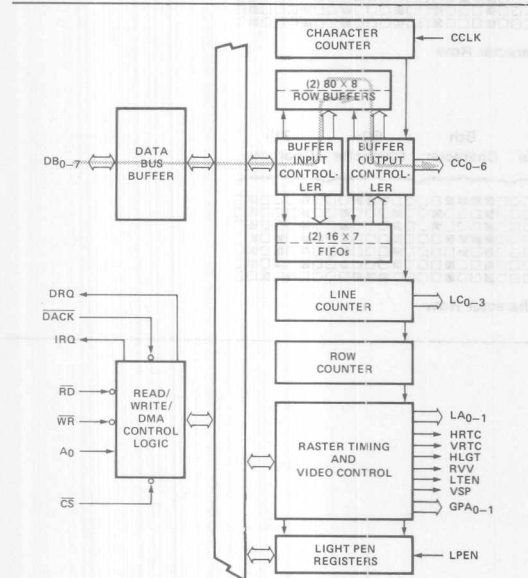


Figure 6. Second Buffer Filled, First Row Displayed

After all the lines of the character row are scanned, the roles of the two row buffers are reversed and the same procedure is followed for the next row.

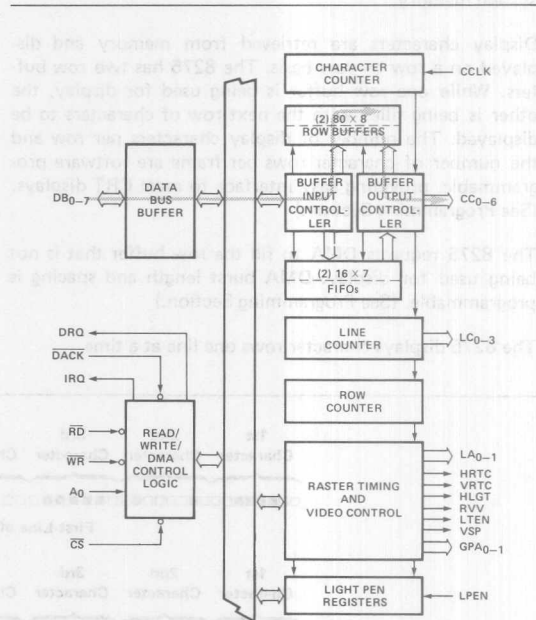


Figure 7. First Buffer Filled with Third Row, Second Row Displayed

This is repeated until all of the character rows are displayed.

## Display Format

### Screen Format

The 8275 can be programmed to generate from 1 to 80 characters per row, and from 1 to 64 rows per frame.

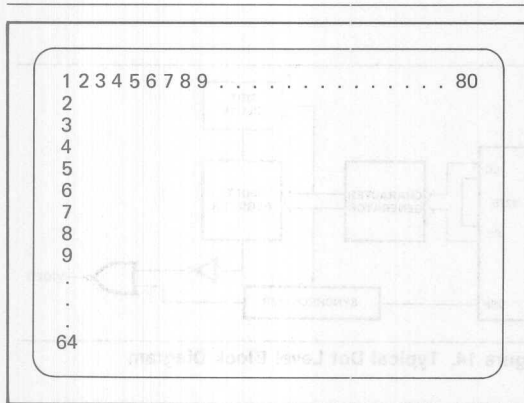


Figure 8. Screen Format

The 8275 can also be programmed to blank alternate rows. In this mode, the first row is displayed, the second blanked, the third displayed, etc. DMA is not requested for the blanked rows.

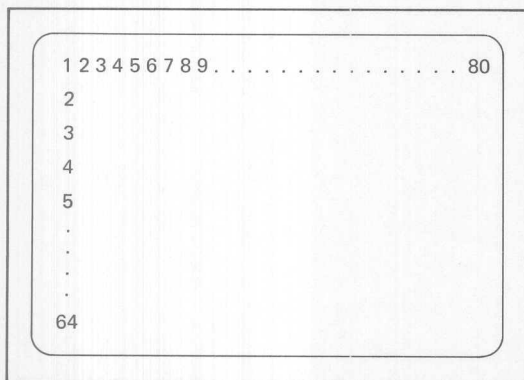


Figure 9. Blank Alternate Rows Mode

### Row Format

The 8275 is designed to hold the line count stable while outputting the appropriate character codes during each horizontal sweep. The line count is incremented during horizontal retrace and the whole row of character codes are output again during the next sweep. This is continued until the whole character row is displayed.

The number of lines (horizontal sweeps) per character row is programmable from 1 to 16.

The output of the line counter can be programmed to be in one of two modes.

In mode 0, the output of the line counter is the same as the line number.

In mode 1, the line counter is offset by one from the line number.

**Note:** In mode 1, while the first line (line number 0) is being displayed, the last count is output by the line counter (see examples).

Line Number		Line Counter Mode 0	Line Counter Mode 1
0	□ □ □ □ □ □ □ □	0000	1111
1	□ □ □ □ ■ □ □ □	0001	0000
2	□ □ ■ □ □ □ □ □	0010	0001
3	□ □ ■ □ □ □ □ □	0011	0010
4	□ ■ □ □ □ □ □ □	0100	0011
5	□ ■ □ □ □ □ □ □	0101	0100
6	□ ■ ■ □ □ □ □ □	0110	0101
7	□ ■ □ □ □ □ □ □	0111	0110
8	□ ■ □ □ □ □ □ □	1000	0111
9	□ ■ □ □ □ □ □ □	1001	1000
10	□ □ □ □ □ □ □ □	1010	1001
11	□ □ □ □ □ □ □ □	1011	1010
12	□ □ □ □ □ □ □ □	1100	1011
13	□ □ □ □ □ □ □ □	1101	1100
14	□ □ □ □ □ □ □ □	1110	1101
15	□ □ □ □ □ □ □ □	1111	1110

Figure 10. Example of a 16-Line Format

Line Number		Line Counter Mode 0	Line Counter Mode 1
0	□ □ □ □ □ □ □ □	0000	1001
1	□ □ □ □ ■ □ □ □	0001	0000
2	□ □ ■ □ □ □ □ □	0010	0001
3	□ ■ □ □ □ □ □ □	0011	0010
4	□ ■ □ □ □ □ □ □	0100	0011
5	□ ■ ■ □ □ □ □ □	0101	0100
6	□ ■ □ □ □ □ □ □	0110	0101
7	□ ■ □ □ □ □ □ □	0111	0110
8	□ □ □ □ □ □ □ □	1000	0111
9	□ □ □ □ □ □ □ □	1001	1000

Figure 11. Example of a 10-Line Format

Mode 0 is useful for character generators that leave address zero blank and start at address 1. Mode 1 is useful for character generators which start at address zero.

ber 0 to 15). This is independent of the line counter mode.

If the line number of the underline is greater than 7 (line number MSB = 1), then the top and bottom lines will be blanked.

Line Number		Line Counter Mode 0	Line Counter Mode 1
0	□ □ □ □ □ □ □ □	0000	1011
1	□ □ □ □ ■ □ □ □	0001	0000
2	□ □ □ □ ■ □ □ □	0010	0001
3	□ □ ■ □ □ □ □ □	0011	0010
4	□ ■ □ □ □ □ □ □	0100	0011
5	□ ■ □ □ □ □ □ □	0101	0100
6	□ ■ ■ □ □ □ □ □	0110	0101
7	□ ■ □ □ □ □ □ □	0111	0110
8	□ ■ □ □ □ □ □ □	1000	0111
9	□ ■ □ □ □ □ □ □	1001	1000
10	■ ■ ■ ■ ■ ■ ■ ■	1010	1001
11	□ □ □ □ □ □ □ □	1011	1010

Top and Bottom Lines are Blanked

Figure 12. Underline in Line Number 10

If the line number of the underline is less than or equal to 7 (line number MSB = 0), then the top and bottom lines will not be blanked.

Line Number		Line Counter Mode 0	Line Counter Mode 1
0	□ □ □ ■ □ □ □ □	0000	0111
1	□ □ ■ □ ■ □ □ □	0001	0000
2	□ ■ □ □ □ □ □ □	0010	0001
3	□ ■ □ □ □ □ □ □	0011	0010
4	□ ■ ■ □ □ □ □ □	0100	0011
5	□ ■ □ □ □ □ □ □	0101	0100
6	□ ■ □ □ □ □ □ □	0110	0101
7	■ ■ ■ ■ ■ ■ ■ ■	0111	0110

Top and Bottom Lines are not Blanked

Figure 13. Underline in Line Number 7

If the line number of the underline is greater than the maximum number of lines, the underline will not appear.

Blanking is accomplished by the VSP (Video Suppression) signal. Underline is accomplished by the LTEN (Light Enable) signal.

Dot width and character width are dependent upon the external timing and control circuitry.

Dot level timing circuitry should be designed to accept the parallel output of the character generator and shift it out serially at the rate required by the CRT display.

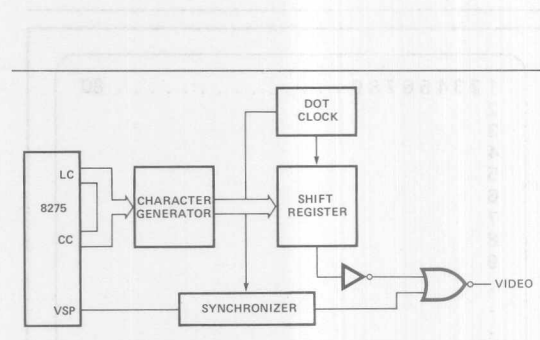


Figure 14. Typical Dot Level Block Diagram

Dot width is a function of dot clock frequency.

Character width is a function of the character generator width.

Horizontal character spacing is a function of the shift register length.

**Note:** Video control and timing signals must be synchronized with the video signal due to the character generator access delay.

### Raster Timing

The character counter is driven by the character clock input (CCLK). It counts out the characters being displayed (programmable from 1 to 80). It then causes the line counter to increment, and it starts counting out the horizontal retrace interval (programmable from 2 to 32). This is constantly repeated.

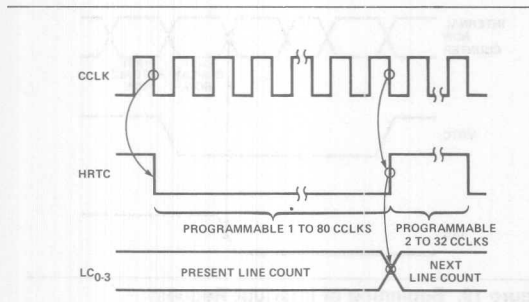


Figure 15. Line Timing

The line counter is driven by the character counter. It is used to generate the line address outputs (LC<sub>0-3</sub>) for the character generator. After it counts all of the lines in a character row (programmable from 1 to 16), it increments the row counter, and starts over again. (See Character Format Section for detailed description of Line Counter functions.)

The row counter is an internal counter driven by the line counter. It controls the functions of the row buffers and counts the number of character rows displayed.

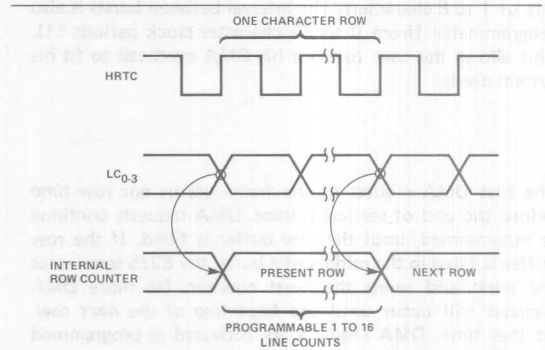


Figure 16. Row Timing

After the row counter counts all of the rows in a frame (programmable from 1 to 64), it starts counting out the vertical retrace interval (programmable from 1 to 4).

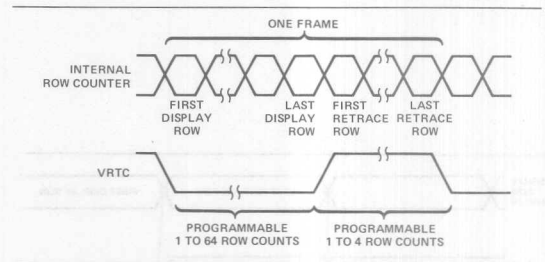


Figure 17. Frame Timing

The Video Suppression Output (VSP) is active during horizontal and vertical retrace intervals.

Dot level timing circuitry must synchronize these outputs with the video signal to the CRT Display.

### DMA Timing

The 8275 can be programmed to request burst DMA transfers of 1 to 8 characters. The interval between bursts is also programmable (from 0 to 55 character clock periods  $\pm 1$ ). This allows the user to tailor his DMA overhead to fit his system needs.

The first DMA request of the frame occurs one row time before the end of vertical retrace. DMA requests continue as programmed, until the row buffer is filled. If the row buffer is filled in the middle of a burst, the 8275 terminates the burst and resets the burst counter. No more DMA requests will occur until the *beginning* of the next row. At that time, DMA requests are activated as programmed until the other buffer is filled.

If, for any reason, there is a DMA underrun, a flag in the status word will be set.

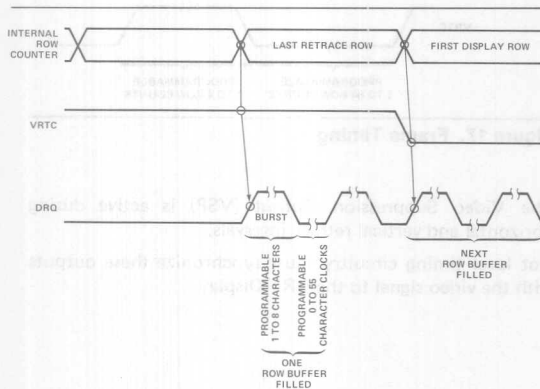


Figure 18. DMA Timing

The DMA controller is typically initialized for the next frame at the end of the current frame.

### Interrupt Timing

The 8275 can be programmed to generate an interrupt request at the end of each frame. This can be used to reinitialize the DMA controller. If the 8275 interrupt enable flag is set, an interrupt request will occur at the *beginning* of the last display row.

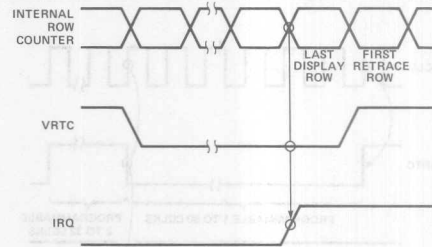


Figure 19. Beginning of Interrupt Request

IRQ will go inactive after the status register is read.



Figure 20. End of Interrupt Request

A reset command will also cause IRQ to go inactive, but this is not recommended during normal service.

Another method of reinitializing the DMA controller is to have the DMA controller itself interrupt on terminal count. With this method, the 8275 interrupt enable flag should not be set.

**Note:** Upon power-up, the 8275 Interrupt Enable Flag may be set. As a result, the user's cold start routine should write a reset command to the 8275 before system interrupts are enabled.

## VISUAL ATTRIBUTES AND SPECIAL CODES

The characters processed by the 8275 are 8-bit quantities. The character code outputs provide the character generator with 7 bits of address. The Most Significant Bit is the extra bit and it is used to determine if it is a normal display character (MSB = 0), or if it is a Visual Attribute or Special Code (MSB = 1).

There are two types of Visual Attribute Codes. They are Character Attributes and Field Attributes.

### Character Attribute Codes

Character attribute codes are codes that can be used to generate graphics symbols without the use of a character generator. This is accomplished by selectively activating the Line Attribute outputs (LA<sub>0-1</sub>), the Video Suppression output (VSP), and the Light Enable output. The dot level timing circuitry can use these signals to generate the proper symbols.

Character attributes can be programmed to blink or be highlighted individually. Blinking is accomplished with the Video Suppression output (VSP). Blink frequency is equal to the screen refresh frequency divided by 32. Highlighting is accomplished by activating the Highlight output (HGLT).

### Character Attributes

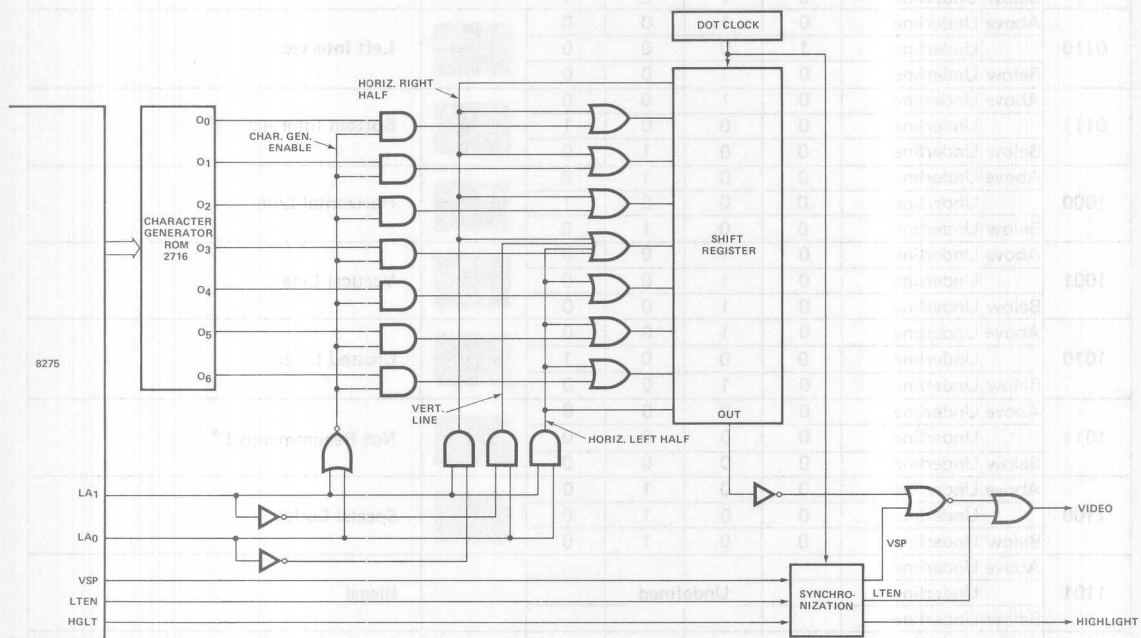
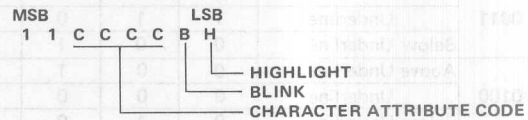


Figure 21. Typical Character Attribute Logic

CHARACTER ATTRIBUTE CODE "CCCC"		OUTPUTS				SYMBOL	DESCRIPTION
		LA <sub>1</sub>	LA <sub>0</sub>	VSP	LTEN		
0000	Above Underline	0	0	1	0		Top Left Corner
	Underline	1	0	0	0		
	Below Underline	0	1	0	0		
0001	Above Underline	0	0	1	0		Top Right Corner
	Underline	1	1	0	0		
	Below Underline	0	1	0	0		
0010	Above Underline	0	1	0	0		Bottom Left Corner
	Underline	1	0	0	0		
	Below Underline	0	0	1	0		
0011	Above Underline	0	1	0	0		Bottom Right Corner
	Underline	1	1	0	0		
	Below Underline	0	0	1	0		
0100	Above Underline	0	0	1	0		Top Intersect
	Underline	0	0	0	1		
	Below Underline	0	1	0	0		
0101	Above Underline	0	1	0	0		Right Intersect
	Underline	1	1	0	0		
	Below Underline	0	1	0	0		
0110	Above Underline	0	1	0	0		Left Intersect
	Underline	1	0	0	0		
	Below Underline	0	1	0	0		
0111	Above Underline	0	1	0	0		Bottom Intersect
	Underline	0	0	0	1		
	Below Underline	0	0	1	0		
1000	Above Underline	0	0	1	0		Horizontal Line
	Underline	0	0	0	1		
	Below Underline	0	0	1	0		
1001	Above Underline	0	1	0	0		Vertical Line
	Underline	0	1	0	0		
	Below Underline	0	1	0	0		
1010	Above Underline	0	1	0	0		Crossed Lines
	Underline	0	0	0	1		
	Below Underline	0	1	0	0		
1011	Above Underline	0	0	0	0		Not Recommended *
	Underline	0	0	0	0		
	Below Underline	0	0	0	0		
1100	Above Underline	0	0	1	0		Special Codes
	Underline	0	0	1	0		
	Below Underline	0	0	1	0		
1101	Above Underline						Illegal
	Underline		Undefined				
	Below Underline						
1110	Above Underline						Illegal
	Underline		Undefined				
	Below Underline						
1111	Above Underline						Illegal
	Underline		Undefined				
	Below Underline						

\*Character Attribute Code 1011 is not recommended for normal operation. Since none of the attribute outputs are active, the character Generator will not be disabled, and an indeterminate character will be generated.

Character Attribute Codes 1101, 1110, and 1111 are illegal.

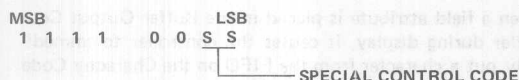
Blinking is active when B = 1.

Highlight is active when H = 1.

## Special Codes

Four special codes are available to help reduce memory, software, or DMA overhead.

### Special Control Character



S	S	FUNCTION
0	0	End of Row
0	1	End of Row-Stop DMA
1	0	End of Screen
1	1	End of Screen-Stop DMA

The End of Row Code (00) activates VSP and holds it to the end of the line.

The End of Row-Stop DMA Code (01) causes the DMA Control Logic to stop DMA for the rest of the row when it is written into the Row Buffer. It affects the display in the same way as the End of Row Code (00).

The End of Screen Code (10) activates VSP and holds it to the end of the frame.

The End of Screen-Stop DMA Code (11) causes the DMA Control Logic to stop DMA for the rest of the frame when it is written into the Row Buffer. It affects the display in the same way as the End of Screen Code (10).

If the Stop DMA feature is not used, all characters after an End of Row character are ignored, except for the End of Screen character, which operates normally. All characters after an End of Screen character are ignored.

**Note:** If a Stop DMA character is not the last character in a burst or row, DMA is not stopped until after the next character is read. In this situation, a dummy character must be placed in memory after the Stop DMA character.

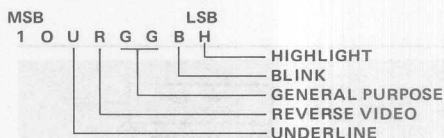
## Field Attributes

The field attributes are control codes which affect the visual characteristics for a field of characters, starting at the character following the code up to, and including, the character which precedes the *next* field attribute code, or up to the end of the frame. The field attributes are reset during the vertical retrace interval.

There are six field attributes:

1. **Blink** — Characters following the code are caused to blink by activating the Video Suppression output (VSP). The blink frequency is equal to the screen refresh frequency divided by 32.
2. **Highlight** — Characters following the code are caused to be highlighted by activating the Highlight output (HGLT).
3. **Reverse Video** — Characters following the code are caused to appear with reverse video by activating the Reverse Video output (RVV).
4. **Underline** — Characters following the code are caused to be underlined by activating the Light Enable output (LTEN).
- 5,6. **General Purpose** — There are two additional 8275 outputs which act as general purpose, independently programmable field attributes.  $GPA_{0-1}$  are active high outputs.

### Field Attribute Code



H = 1 FOR HIGHLIGHTING  
B = 1 FOR BLINKING  
R = 1 FOR REVERSE VIDEO  
U = 1 FOR UNDERLINE  
GG =  $GPA_1$ ,  $GPA_0$

The 8275 can be programmed to provide visible or invisible field attribute characters.

If the 8275 is programmed in the visible field attribute mode, all field attributes will occupy a position on the screen. They will appear as blanks caused by activation of the Video Suppression output (VSP). The chosen visual attributes are activated after this blanked character.

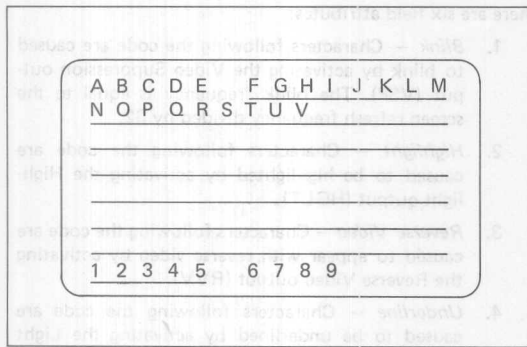


Figure 22. Example of the Visible Field Attribute Mode (Underline Attribute)

If the 8275 is programmed in the invisible field attribute mode, the 8275 FIFO is activated.

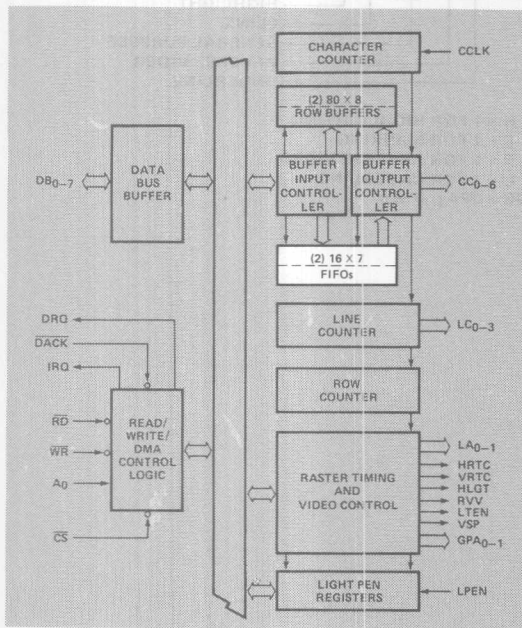


Figure 23. Block Diagram Showing FIFO Activation

Each row buffer has a corresponding FIFO. These FIFOs are 16 characters by 7 bits in size.

When a field attribute is placed in the row buffer during DMA, the buffer input controller recognizes it and places the *next* character in the proper FIFO.

When a field attribute is placed in the Buffer Output Controller during display, it causes the controller to immediately put a character from the FIFO on the Character Code outputs (CC0-6). The chosen Visual Attributes are also activated.

Since the FIFO is 16 characters long, no more than 16 field attribute characters may be used per line in this mode. If more are used, a bit in the status word is set and the first characters in the FIFO are written over and lost.

**Note:** Since the FIFO is 7 bits wide, the MSB of any characters put in it are stripped off. Therefore, a Visual Attribute or Special Code must *not* immediately follow a field attribute code. If this situation does occur, the Visual Attribute or Special Code will be treated as a normal display character.

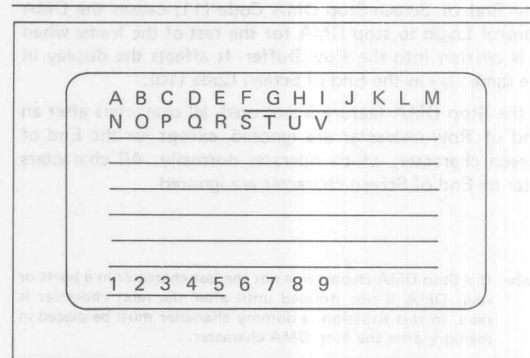


Figure 24. Example of the Invisible Field Attribute Mode (Underline Attribute)

#### Field and Character Attribute Interaction

Character Attribute Symbols are affected by the Reverse Video (RRV) and General Purpose (GPA0-1) field attributes. They are not affected by Underline, Blink or High-light field attributes; however, these characteristics can be programmed *individually* for Character Attribute Symbols.

### Cursor Timing

The cursor location is determined by a cursor row register and a character position register which are loaded by command to the controller. The cursor can be programmed to appear on the display as:

1. a blinking underline
2. a blinking reverse video block
3. a non-blinking underline
4. a non-blinking reverse video block

The cursor blinking frequency is equal to the screen refresh frequency divided by 16.

If a non-blinking reverse video *cursor* appears in a non-blinking reverse video *field*, the cursor will appear as a normal video block.

If a non-blinking underline *cursor* appears in a non-blinking underline *field*, the cursor will not be visible.

### Light Pen Detection

A light pen consists of a micro switch and a tiny light sensor. When the light pen is pressed against the CRT screen, the micro switch enables the light sensor. When the raster sweep reaches the light sensor, it triggers the light pen output.

If the output of the light pen is presented to the 8275 LPEN input, the row and character position coordinates are stored in a pair of registers. These registers can be read on command. A bit in the status word is set, indicating that the light pen signal was detected. The LPEN input must be a 0 to 1 transition for proper operation.

**Note:** Due to internal and external delays, the character position coordinate will be off by at least three character positions. This has to be corrected in software.

### Device Programming

The 8275 has two programming registers, the Command Register (CREG) and the Parameter Register (PREG). It also has a Status Register (SREG). The Command Register can only be written into and the Status Registers can only be read from. They are addressed as follows:

A <sub>0</sub>	OPERATION	REGISTER
0	Read	PREG
0	Write	PREG
1	Read	SREG
1	Write	CREG

The 8275 expects to receive a command and a sequence of 0 to 4 parameters, depending on the command. If the proper number of parameter bytes are not received before another command is given, a status flag is set, indicating an improper command.

### Instruction Set

The 8275 instruction set consists of 8 commands.

COMMAND	NO. OF PARAMETER BYTES
Reset	4
Start Display	0
Stop Display	0
Read Light Pen	2
Load Cursor	2
Enable Interrupt	0
Disable Interrupt	0
Preset Counters	0

In addition, the status of the 8275 (SREG) can be read by the CPU at any time.

Command	Write	1	Reset Command	0 0 0 0 0 0 0 0
	Write	0	Screen Comp Byte 1	S H H H H H H H
	Write	0	Screen Comp Byte 2	V V R R R R R R
	Write	0	Screen Comp Byte 3	U U U U L L L L
	Write	0	Screen Comp Byte 4	M F C C Z Z Z Z

**Action** — After the reset command is written, DMA requests stop, 8275 interrupts are disabled, and the VSP output is used to blank the screen. HRTC and VRTC continue to run. HRTC and VRTC timing are random on power-up.

As parameters are written, the screen composition is defined.

#### Parameter — S Spaced Rows

S	FUNCTIONS
0	Normal Rows
1	Spaced Rows

#### Parameter — HHHHHHH Horizontal Characters/Row

H H H H H H H	NO. OF CHARACTERS PER ROW
0 0 0 0 0 0 0	1
0 0 0 0 0 0 1	2
0 0 0 0 0 1 0	3
.	.
.	.
1 0 0 1 1 1 1	80
1 0 1 0 0 0 0	Undefined
.	.
.	.
1 1 1 1 1 1 1	Undefined

#### Parameter — VV Vertical Retrace Row Count

V V	NO. OF ROW COUNTS PER VRTC
0 0	1
0 1	2
1 0	3
1 1	4

#### Parameter — RRRRRR Vertical Rows/Frame

R R R R R R	NO. OF ROWS/FRAME
0 0 0 0 0 0	1
0 0 0 0 0 1	2
0 0 0 0 1 0	3
.	.
.	.
1 1 1 1 1 1	64

U U U U	UNDERLINE
0 0 0 0	1
0 0 0 1	2
0 0 1 0	3
.	.
.	.
1 1 1 1	16

#### Parameter — LLLL Number of Lines per Character Row

L L L L	NO. OF LINES/ROW
0 0 0 0	1
0 0 0 1	2
0 0 1 0	3
.	.
.	.
1 1 1 1	16

#### Parameter — M Line Counter Mode

M	LINE COUNTER MODE
0	Mode 0 (Non-Offset)
1	Mode 1 (Offset by 1 Count)

#### Parameter — F Field Attribute Mode

F	FIELD ATTRIBUTE MODE
0	Transparent
1	Non-Transparent

#### Parameter — CC Cursor Format

C C	CURSOR FORMAT
0 0	Blinking reverse video block
0 1	Blinking underline
1 0	Nonblinking reverse video block
1 1	Nonblinking underling

#### Parameter — ZZZZ Horizontal Retrace Count

Z Z Z Z	NO. OF CHARACTER COUNTS PER HRTC
0 0 0 0	2
0 0 0 1	4
0 0 1 0	6
.	.
.	.
1 1 1 1	32

**Note:** uuuu MSB determines blanking of top and bottom lines (1 = blanked, 0 = not blanked).

## 2. Start Display Command:

	OPERATION	A <sub>0</sub>	DESCRIPTION	MSB	DATA BUS	LSB
Command	Write	1	Start Display	0	0	1 S S S B B
No parameters						

## S S S BURST SPACE CODE

S S S	NO. OF CHARACTER CLOCKS BETWEEN DMA REQUESTS
0 0 0	0
0 0 1	7
0 1 0	15
0 1 1	23
1 0 0	31
1 0 1	39
1 1 0	47
1 1 1	55

## B B BURST COUNT CODE

B B	NO. OF DMA CYCLES PER BURST
0 0	1
0 1	2
1 0	4
1 1	8

**Action** — 8275 interrupts are enabled, DMA requests begin, video is enabled, Interrupt Enable and Video Enable status flags are set.

## 3. Stop Display Command:

	OPERATION	A <sub>0</sub>	DESCRIPTION	MSB	DATA BUS	LSB
Command	Write	1	Stop Display	0	1	0 0 0 0 0 0
No parameters						

**Action** — Disables video, interrupts remain enabled, HRTC and VRTC continue to run, Video Enable status flag is reset, and the "Start Display" command must be given to re-enable the display.

## 4. Read Light Pen Command

	OPERATION	A <sub>0</sub>	DESCRIPTION	MSB	DATA BUS	LSB
Command	Write	1	Read Light Pen	0	1	1 0 0 0 0 0
Parameters	Read	0	Char. Number	(Char. Position in Row)		
	Read	0	Row Number	(Row Number)		

**Action** — The 8275 is conditioned to supply the contents of the light pen position registers in the next two read cycles of the parameter register. Status flags are not affected.

**Note:** Software correction of light pen position is required.

## 5. Load Cursor Position:

	OPERATION	A <sub>0</sub>	DESCRIPTION	MSB	DATA BUS	LSB
Command	Write	1	Load Cursor	1	0	0 0 0 0 0 0
Parameters	Write	0	Char. Number	(Char. Position in Row)		
	Write	0	Row Number	(Row Number)		

**Action** — The 8275 is conditioned to place the next two parameter bytes into the cursor position registers. Status flags not affected.

## 6. Enable Interrupt Command:

	OPERATION	A <sub>0</sub>	DESCRIPTION	MSB	DATA BUS	LSB
Command	Write	1	Enable Interrupt	1	0	1 0 0 0 0 0
No parameters						

**Action** — The interrupt enable status flag is set and interrupts are enabled.

## 7. Disable Interrupt Command:

	OPERATION	A <sub>0</sub>	DESCRIPTION	MSB	DATA BUS	LSB
Command	Write	1	Disable Interrupt	1	1	0 0 0 0 0 0
No parameters						

**Action** — Interrupts are disabled and the interrupt enable status flag is reset.

## 8. Preset Counters Command:

	OPERATION	A <sub>0</sub>	DESCRIPTION	MSB	DATA BUS	LSB
Command	Write	1	Preset Counters	1	1	1 0 0 0 0 0
No parameters						

**Action** — The internal timing counters are preset, corresponding to a screen display position at the top left corner. Two character clocks are required for this operation. The counters will remain in this state until any other command is given.

This command is useful for system debug and synchronization of clustered CRT displays on a single CPU.

## Status Flags

	OPERATION	A <sub>0</sub>	DESCRIPTION	DATA BUS	
				MSB	LSB
Command	Read	1	Status Word	0 IE IR LP IC VE OU FO	

IE — (Interrupt Enable) Set or reset by command. It enables vertical retrace interrupt. It is automatically set by a "Start Display" command and reset with the "Reset" command.

IR — (Interrupt Request) This flag is set at the beginning of display of the last row of the frame if the interrupt enable flag is set. It is reset after a status read operation.

LP — This flag is set when the light pen input (LPEN) is activated and the light pen registers have been loaded. This flag is automatically reset after a status read.

IC — (Improper Command) This flag is set when a command parameter string is too long or too short. The flag is automatically reset after a status read.

VE — (Video Enable) This flag indicates that video operation of the CRT is enabled. This flag is set on a "Start Display" command, and reset on a "Stop Display" or "Reset" command.

DU — (DMA Underrun) This flag is set whenever a data underrun occurs during DMA transfers. Upon detection of DU, the DMA operation is stopped and the screen is blanked until after the vertical retrace interval. This flag is reset after a status read.

FO — (FIFO Overrun) This flag is set whenever the FIFO is overrun. It is reset on a status read.

**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature Under Bias . . . . . 0°C to 70°C  
 Storage Temperature . . . . . -65°C to +150°C  
 Voltage On Any Pin  
     With Respect to Ground . . . . . -0.5V to +7V  
 Power Dissipation . . . . . 1 Watt

\*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

**D.C. CHARACTERISTICS**

T<sub>A</sub> = 0°C to 70°C; V<sub>CC</sub> = 5V ±5%

SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
V <sub>IL</sub>	Input Low Voltage	-0.5	0.8	V	
V <sub>IH</sub>	Input High Voltage	2.0	V <sub>CC</sub> +0.5V	V	
V <sub>OL</sub>	Output Low Voltage		0.45	V	I <sub>OL</sub> = 2.2 mA
V <sub>OH</sub>	Output High Voltage	2.4		V	I <sub>OH</sub> = -400 μA
I <sub>IL</sub>	Input Load Current		±10	μA	V <sub>IN</sub> = V <sub>CC</sub> to 0V
I <sub>OFL</sub>	Output Float Leakage		±10	μA	V <sub>OUT</sub> = V <sub>CC</sub> to 0V
I <sub>CC</sub>	V <sub>CC</sub> Supply Current		160	mA	

**CAPACITANCE**

T<sub>A</sub> = 25°C; V<sub>CC</sub> = GND = 0V

SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
C <sub>IN</sub>	Input Capacitance		10	pF	f <sub>c</sub> = 1 MHz
C <sub>I/O</sub>	I/O Capacitance		20	pF	Unmeasured pins returned to V <sub>SS</sub> .

## Bus Parameters (Note 1)

### Read Cycle:

SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
$t_{AR}$	Address Stable Before READ	0		ns	
$t_{RA}$	Address Hold Time for READ	0		ns	
$t_{RR}$	READ Pulse Width	250		ns	
$t_{RD}$	Data Delay from READ		200	ns	$C_L = 150 \text{ pF}$
$t_{DF}$	READ to Data Floating	20	100	ns	

### Write Cycle:

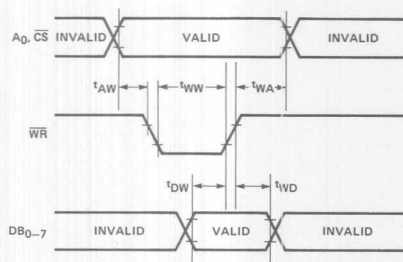
SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
$t_{AW}$	Address Stable Before WRITE	0		ns	
$t_{WA}$	Address Hold Time for WRITE	0		ns	
$t_{WW}$	WRITE Pulse Width	250		ns	
$t_{DW}$	Data Setup Time for WRITE	150		ns	
$t_{WD}$	Data Hold Time for WRITE	0		ns	

### Clock Timing:

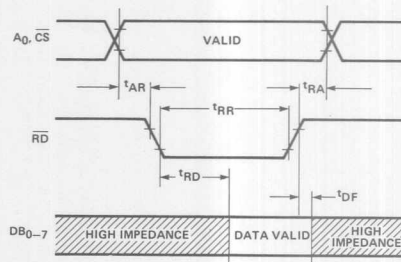
SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
$t_{CLK}$	Clock Period	320		ns	
$t_{KH}$	Clock High	120		ns	
$t_{KL}$	Clock Low	120		ns	
$t_{KR}$	Clock Rise	5	30	ns	
$t_{KF}$	Clock Fall	5	30	ns	

Note 1: AC timings measured at  $V_{OH} = 2.0$ ,  $V_{OL} = 0.8$

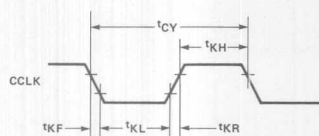
### Write Timing



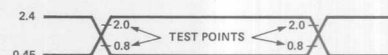
### Read Timing



### Clock Timing



### Input Waveforms (For A.C. Tests)



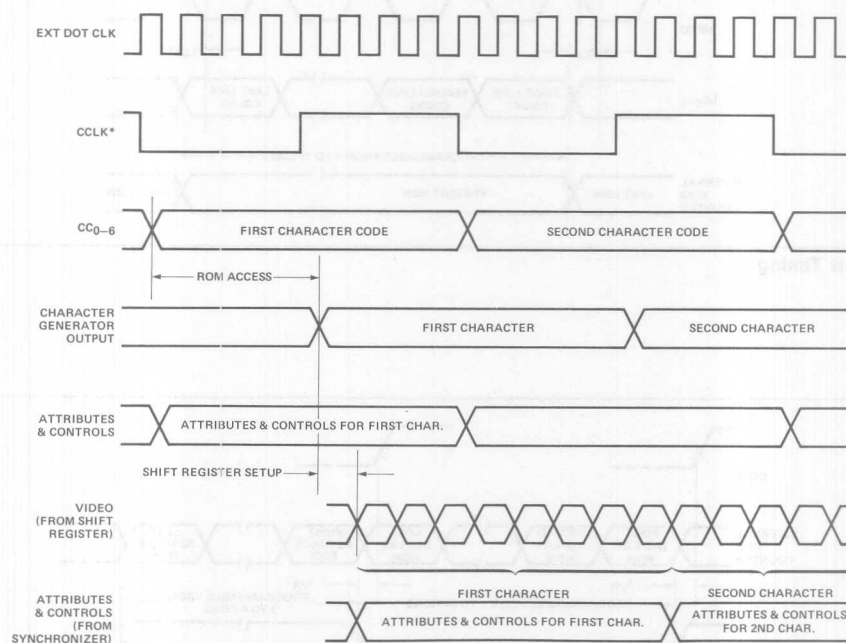
**PRELIMINARY**  
 Notice: This is not a final specification. Some  
 parametric limits are subject to change.

### Other Timing:

SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
$t_{CC}$	Character Code Output Delay		150	ns	$C_L = 50 \text{ pF}$
$t_{HR}$	Horizontal Retrace Output Delay		150	ns	$C_L = 50 \text{ pF}$
$t_{LC}$	Line Count Output Delay		250	ns	$C_L = 50 \text{ pF}$
$t_{AT}$	Control/Attribute Output Delay		250	ns	$C_L = 50 \text{ pF}$
$t_{VR}$	Vertical Retrace Output Delay		250	ns	$C_L = 50 \text{ pF}$
$t_{IR}$	$IRQ \uparrow$ from $CCLK \downarrow$		250	ns	$C_L = 50 \text{ pF}$
$t_{RI}$	$IRQ \downarrow$ from $Rd \uparrow$		250	ns	$C_L = 50 \text{ pF}$
$t_{KQ}$	$DRQ \uparrow$ from $CCLK \downarrow$		250	ns	$C_L = 50 \text{ pF}$
$t_{WQ}$	$DRQ \uparrow$ from $WR \uparrow$		250	ns	$C_L = 50 \text{ pF}$
$t_{RQ}$	$DRQ \downarrow$ from $WR \downarrow$		250	ns	$C_L = 50 \text{ pF}$
$t_{LR}$	$DACK \downarrow$ to $WR \downarrow$	0		ns	
$t_{RL}$	$WR \uparrow$ to $DACK \uparrow$	0		ns	
$t_{PR}$	LPEN Rise		50	ns	
$t_{PH}$	LPEN Hold	100		ns	

**Note:** Timing measurements are made at the following reference voltages: Output "1" = 2.0V, "0" = 0.8V.

### WAVEFORMS



\*CCLK IS A MULTIPLE OF THE DOT CLOCK AND AN INPUT TO THE 8275.

**Figure 25. Typical Dot Level Timing**

**PRELIMINARY**  
 Note: This is not a final specification. Some  
 parameters are subject to change.

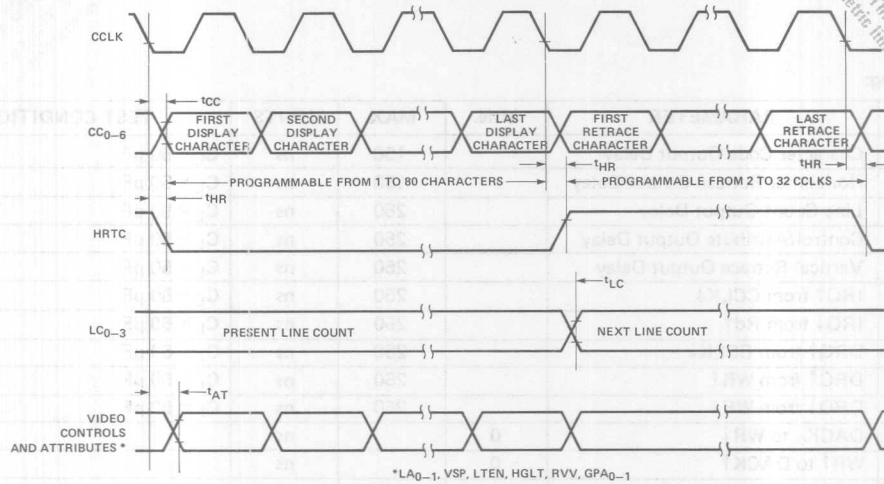


Figure 26. Line Timing

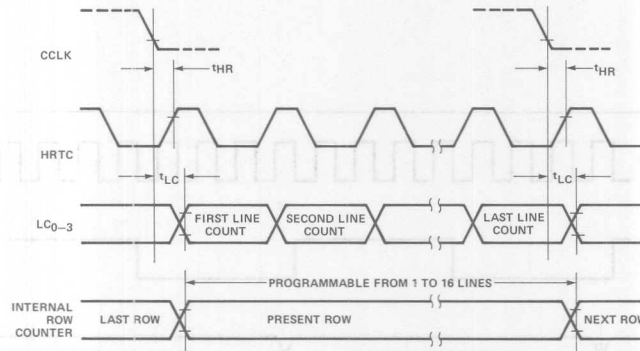


Figure 27. Row Timing

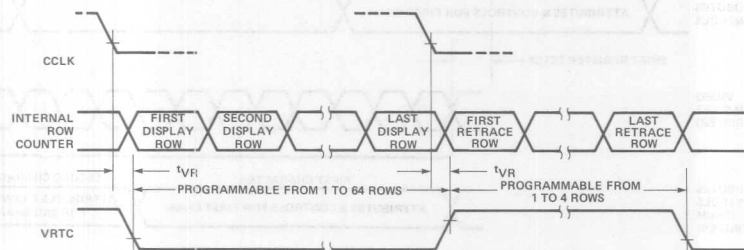


Figure 28. Frame Timing

**PRELIMINARY**  
 Notice: This is not a final specification. Some parametric limits are subject to change.

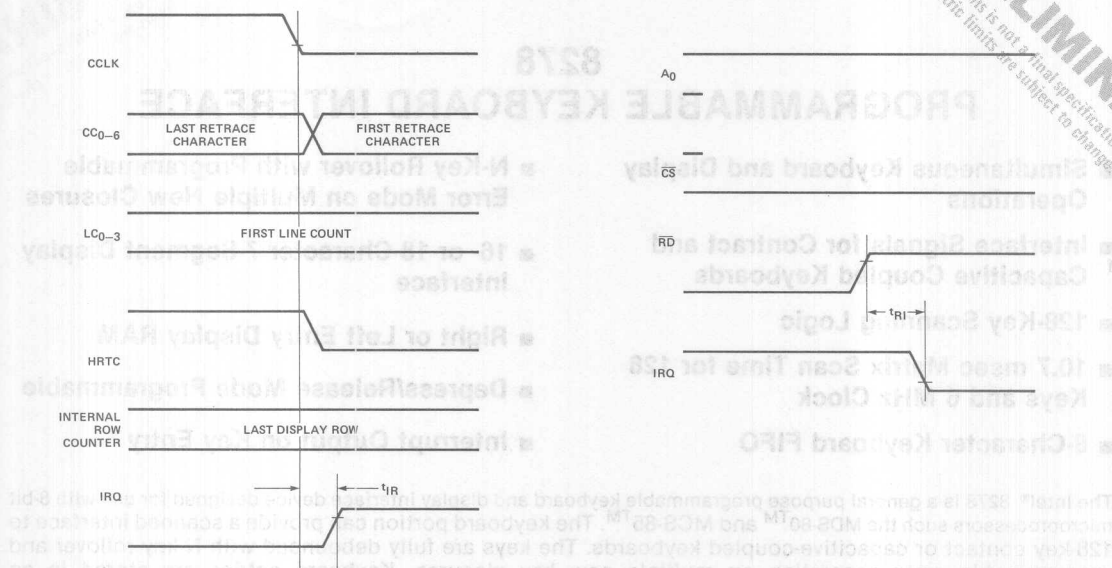


Figure 29. Interrupt Timing

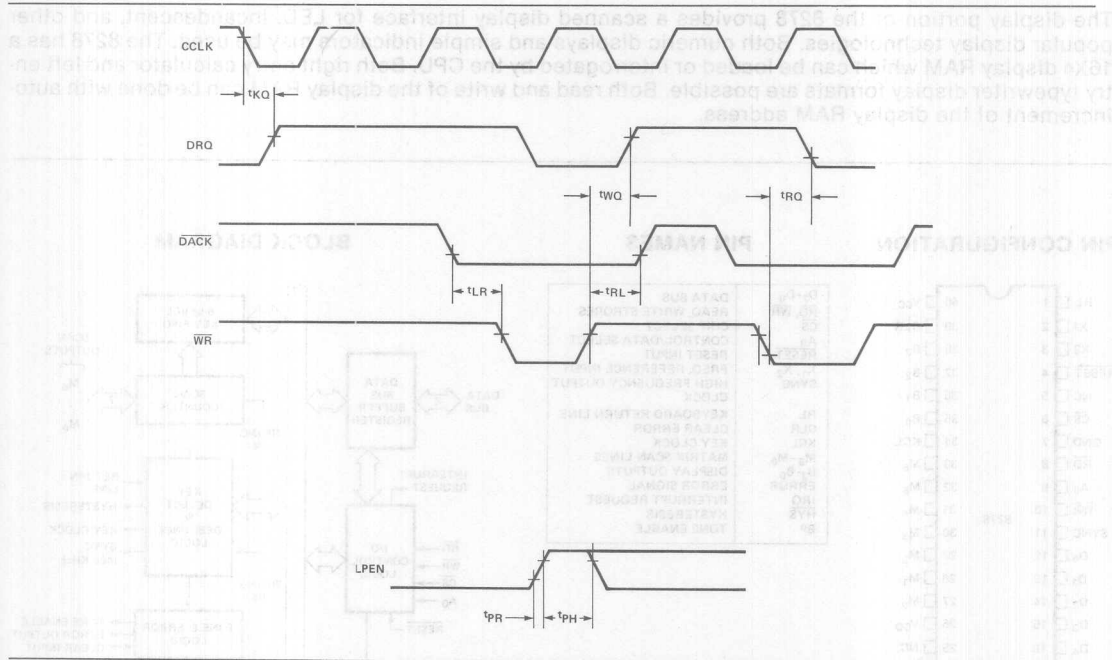


Figure 30. DMA Timing

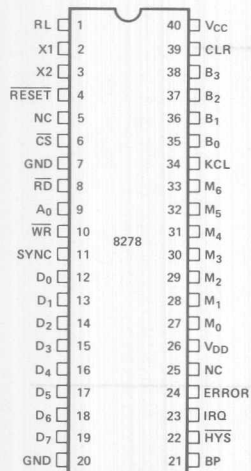
# 8278 PROGRAMMABLE KEYBOARD INTERFACE

- Simultaneous Keyboard and Display Operations
- Interface Signals for Contract and Capacitive Coupled Keyboards
- 128-Key Scanning Logic
- 10.7 msec Matrix Scan Time for 128 Keys and 6 MHz Clock
- 8-Character Keyboard FIFO
- N-Key Rollover with Programmable Error Mode on Multiple New Closures
- 16- or 18-Character 7-Segment Display Interface
- Right or Left Entry Display RAM
- Depress/Release Mode Programmable
- Interrupt Output on Key Entry

The Intel® 8278 is a general purpose programmable keyboard and display interface device designed for use with 8-bit microprocessors such as the MDS-80™ and MCS-85™. The keyboard portion can provide a scanned interface to 128-key contact or capacitive-coupled keyboards. The keys are fully debounced with N-key rollover and programmable error generation on multiple new key closures. Keyboard entries are stored in an 8-character FIFO with overrun status indication when more than 8 characters are entered. Key entries set an interrupt request output to the master CPU.

The display portion of the 8278 provides a scanned display interface for LED, incandescent, and other popular display technologies. Both numeric displays and simple indicators may be used. The 8278 has a 16X4 display RAM which can be loaded or interrogated by the CPU. Both right entry calculator and left entry typewriter display formats are possible. Both read and write of the display RAM can be done with auto-increment of the display RAM address.

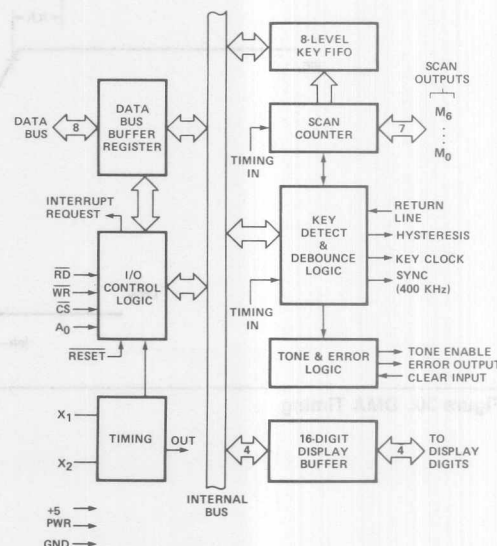
## PIN CONFIGURATION



## PIN NAMES

D <sub>7</sub> -D <sub>0</sub>	DATA BUS
RD, WR	READ, WRITE STROBES
CS	CHIP SELECT
A <sub>0</sub>	CONTROL/DATA SELECT
RESET	RESET INPUT
X <sub>1</sub> , X <sub>2</sub>	FREQ. REFERENCE INPUT
SYNC	HIGH FREQUENCY OUTPUT
CL	CLOCK
RL	KEYBOARD RETURN LINE
CLR	CLEAR ERROR
KCL	KEY CLOCK
M <sub>6</sub> -M <sub>0</sub>	MATRIX SCAN LINES
B <sub>3</sub> -B <sub>0</sub>	DISPLAY OUTPUTS
ERR	ERROR SIGNAL
IRQ	ERROR SIGNAL
HYS	HYSTERESIS
BP	tone ENABLE

## BLOCK DIAGRAM



## PIN DESCRIPTION

The 8278 is packaged in a 40-pin DIP. The following is a brief functional description of each pin.

Signal	Pin No.	Description
D <sub>0</sub> -D <sub>7</sub>	12-19	Three-state, bi-directional data bus lines used to transfer data and commands between the CPU and the 8278.
$\overline{WR}$	10	Write strobe which enables the master CPU to write data and commands between the CPU and the 8278.
$\overline{RD}$	8	Read strobe which enables the master CPU to read data and status from the 8278 internal registers.
$\overline{CS}$	6	Chip select input used to enable reading and writing to the 8278.
A <sub>0</sub>	9	Address input used by the CPU to indicate control or data.
$\overline{RESET}$	4	A low signal on this pin resets the 8278.
X <sub>1</sub> , X <sub>2</sub>	2,3	Inputs for crystal, L-C or external timing signal to determine internal oscillator frequency.
IRQ	23	Interrupt Request Output to the master CPU. In the keyboard mode the IRQ line goes low with each FIFO read and returns high if there is still information in the FIFO or an ERROR has occurred.
M <sub>0</sub> -M <sub>6</sub>	27-33	Matrix scan outputs. These outputs control a decoder which scans the key matrix columns and the 16 display digits. Also, the Matrix scan outputs are used to multiplex the return lines from the key matrix.
RL	1	Input from the multiplexer which indicates whether the key currently being scanned is closed.
$\overline{HYS}$	22	Hysteresis output to the analog detector. (Capacitive keyboard configuration). A "0" means the key currently being scanned has already been recorded.
KCL	34	Key clock output to the analog detector (capacitive keyboard configuration) used to reset the detector before scanning a key.
SYNC	11	High frequency (400 KHz) output signal used in the key scan to detect a closed key (capacitive keyboard configuration).
B <sub>0</sub> -B <sub>3</sub>	35-38	These four lines contain binary coded decimal display information synchronized to the keyboard column scan. The outputs are for multiplexed digital displays.

Signal	Pin No.	Description
ERROR	24	Error signal. This line is high whenever two new key closures are detected during a single scan or when too many characters are entered into the keyboard FIFO. It is reset by a system RESET pulse or by a "1" input on the CLR pin or by the CLEAR ERROR command.
CLR	39	Input used to clear an ERROR condition in the 8278.
BP	21	Tone enable output. This line is high for 10ms following a valid key closure; it is set high and remains high during an ERROR condition.
V <sub>CC</sub> , V <sub>DD</sub>	40,26	+5 volt power input: +5V $\pm$ 10%.
GND	20,7	Signal ground.

## PRINCIPLES OF OPERATION

The following is a description of the major elements of the Programmable Keyboard/Display interface device. Refer to the block diagram in Figure 1.

### I/O Control and Data Buffers

The I/O control section uses the  $\overline{CS}$ , A<sub>0</sub>,  $\overline{RD}$ , and  $\overline{WR}$  lines to control data flow to and from the various internal registers and buffers (see Table 1). All data flow to and from the 8278 is enabled by  $\overline{CS}$ . The 8-bits of information being transferred by the CPU is identified by A<sub>0</sub>. A logic one means information is command or status. A logic zero means the information is data.  $\overline{RD}$  and  $\overline{WR}$  determine the direction of data flow through the Data Bus Buffer (DBB). The DBB register is a bi-directional 8-bit buffer register which connects the internal 8278 bus buffer register to the external bus. When the chip is not selected ( $\overline{CS} = 1$ ) the DBB is in the high impedance state. The DBB acts as an input when ( $\overline{RD}$ ,  $\overline{WR}$ ,  $\overline{CS}$ ) = (1, 0, 0) and an output when ( $\overline{RD}$ ,  $\overline{WR}$ ,  $\overline{CS}$ ) = (0, 1, 0).

$\overline{CS}$	A <sub>0</sub>	$\overline{WR}$	$\overline{RD}$	Condition
0	0	1	0	Read DBB Data
0	1	1	0	Read STATUS
0	0	0	1	Write Data to DBB
0	1	0	1	Write Command to DBB
1	X	X	X	Disable 8278 Bus is High Impedance

### Scan Counter

The scan counter provides the timing to scan the keyboard and display. The four MSB's (M<sub>3</sub>-M<sub>6</sub>) scan the display digits and provide column scan to the keyboard via a 4 to 16 decoder. The three LSB's (M<sub>0</sub>-M<sub>2</sub>) are used to multiplex the row return lines into the 8278.

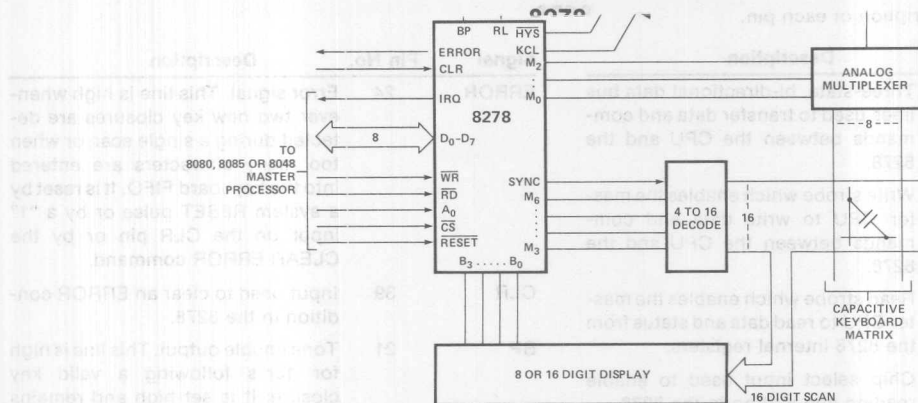


Figure 1. System Configuration for Capacitive-Coupled Keyboard

### Keyboard Debounce and Control

The 8278 system configuration is shown in Figure 2. The rows of the matrix are scanned and the outputs are multiplexed by the 8278. When a key closure is detected, the debounce logic waits about 12 msec to check if the key remains closed. If it does, the address of the key in the matrix is transferred into a FIFO buffer.

### FIFO and FIFO Status

The 8278 contains an 8X8 FIFO character buffer. Each new entry is written into a successive FIFO location and each is then read out in the order of entry. A FIFO status register keeps track of the number of characters in the

FIFO and whether it is full or empty. Too many reads or key entries will be recognized as an error. The status can be read by a  $\overline{RD}$  with  $\overline{CS}$  low and  $A_0$  high. The status logic also provides a  $\overline{IRQ}$  signal to the master processor whenever the FIFO is not empty.

### Display Address Registers and Display RAM

The Display Address registers hold the address of the word currently being written or read by the CPU and the two 4-bit nibbles being displayed. The read/write addresses are programmed by CPU command. They also can be set to auto increment after each read or write. The display RAM can be directly read by the CPU after the correct mode and address is set. Data entry to the display can be set to either left or right entry.

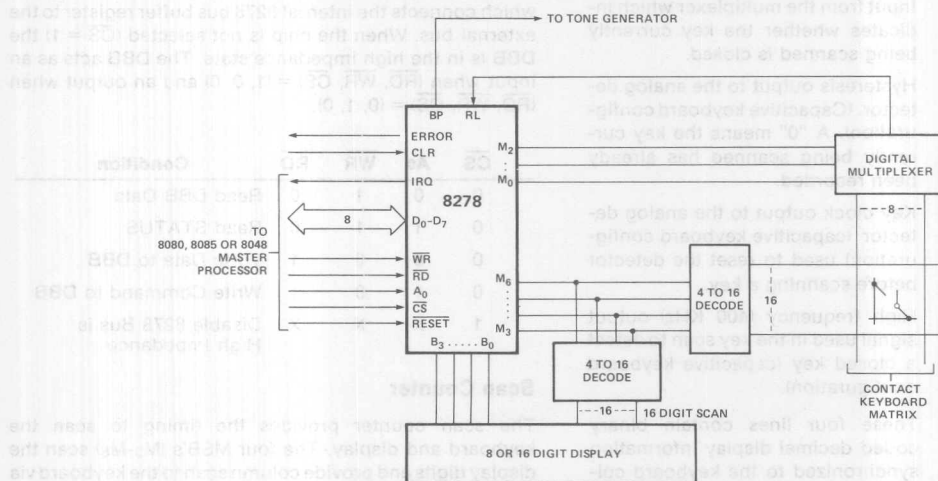
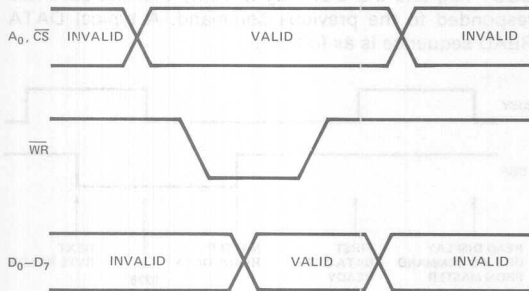


Figure 2. System Configuration for Contact Keyboard

## 8278 COMMANDS

The 8278 operating mode is programmed by the master CPU using the A<sub>0</sub>, WR, and D<sub>0</sub>-D<sub>7</sub> inputs as shown below:



The master CPU presents the proper command on the D<sub>0</sub>-D<sub>7</sub> data lines with A<sub>0</sub>=1 and then sends a WR pulse. The command is latched by the 8278 on the rising edge of the WR and is decoded internally to set the proper operating mode.

## COMMAND SUMMARY

### Keyboard/Display Mode Set

CODE	0	0	0	N	E	I	D	K
------	---	---	---	---	---	---	---	---

where the mode set bits are defined as follows:

- K — the keyboard mode select bit
- 0 — normal key entry mode
- 1 — special function mode: Entry on key closure and on key release
- D — the display entry mode select bit
- 0 — left display entry
- 1 — right display entry
- I — the interrupt request (IRQ) output enable bit
- 0 — enable IRQ output
- 1 — disable IRQ output
- E — the error mode select bit
- 0 — error on multiple key depression
- 1 — no error on multiple key depression
- N — the number of display digits select
- 0 — 16 display digits
- 1 — 8 display digits

NOTE: The default mode following a RESET input is all bits zero:

0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---

### Read FIFO Command

CODE	0	1	0	0	0	0	0	0
------	---	---	---	---	---	---	---	---

### Read Display Command

CODE	0	1	1	AI	A <sub>3</sub>	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>
------	---	---	---	----	----------------	----------------	----------------	----------------

Where AI indicates Auto Increment and A<sub>3</sub>-A<sub>0</sub> is the address of the next display character to be read out.

- AI=1 AUTO increment
- AI=0 no AUTO increment

### Write Display Command

CODE	1	0	0	AI	A <sub>3</sub>	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>
------	---	---	---	----	----------------	----------------	----------------	----------------

Where AI indicates Auto Increment and A<sub>3</sub>-A<sub>0</sub> is the address of the next display character to be written.

### Clear/Blank Command

CODE	1	0	1	UD	BD	CD	CF	CE
------	---	---	---	----	----	----	----	----

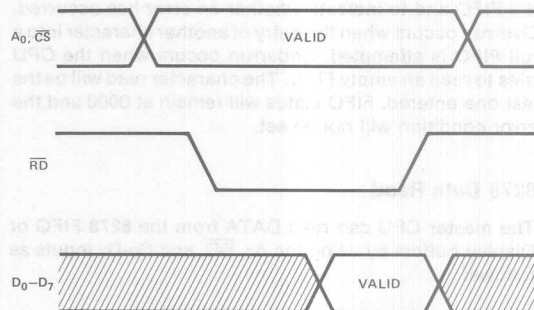
Where the command bits are defined as follows:

- CE = Clear ERROR
- CF = Clear FIFO
- CD = Clear Display to all High
- BD = Blank Display to all High
- UD = Unblank Display

The display is cleared and blanked following a Reset.

### 8278 Status Read

The status register in the 8278 can be read by the master CPU using the A<sub>0</sub>, RD, and D<sub>0</sub>-D<sub>7</sub> inputs as shown below:



The 8278 places 8-bits of status information on the D<sub>0</sub>-D<sub>7</sub> lines following (A<sub>0</sub>, CS, RD) = 1, 0, 0 inputs from the master.

### Status Format

S <sub>3</sub>	S <sub>2</sub>	S <sub>1</sub>	S <sub>0</sub>	B	KE	OBF	IBF
D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>

Where the status bits are defined as follows:

- IBF = Input Buffer Full Flag
- OBF = Output Buffer Full Flag
- KE = Keyboard Error Flag (multiple depression)
- B = BUSY Flag
- S<sub>3</sub>-S<sub>0</sub> = FIFO Status

### Status Description

The S<sub>3</sub>-S<sub>0</sub> status bits indicate the number of entries (0 to 8) in the 8-level FIFO. A FIFO overrun will lock status at 1111. The overrun condition will prevent further key entries until cleared.

A multiple key closure error will set the KE flag and prevent further key entries until cleared.

The IBF and OBF flags signify the status of the 8278 data buffer registers used to transfer information (data, status or commands) to and from the master CPU.

The IBF flag is set when the master CPU writes Data or Commands to the 8278. The IBF flag is cleared by the 8278 during its response to the Data or Command.

The OBF flag is set when the 8278 has output data ready for the master CPU. This flag is cleared by a master CPU Data READ.

The Busy flag in the status register is used as a LOCK-OUT signal to the master processor during response to any command or data write from the master.

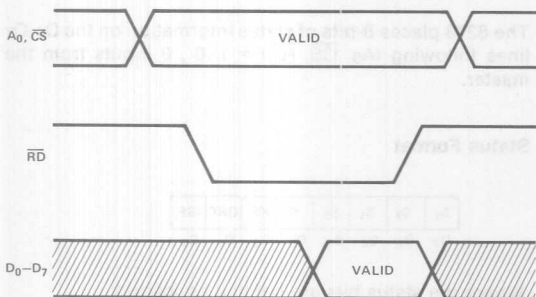
The master must test the Busy flag before each read (during a sequence) to be sure that the 8278 is ready with valid DATA.

The ERROR and TONE outputs from the 8278 are set high for either type of error. Both types of error are cleared by the CLR input, by the CLEAR ERROR command, or by a reset. The FIFO and Display buffers are cleared independently of the Errors.

FIFO status is used to indicate the number of characters in the FIFO and to indicate whether an error has occurred. Overrun occurs when the entry of another character into a full FIFO is attempted. Underrun occurs when the CPU tries to read an empty FIFO. The character read will be the last one entered. FIFO status will remain at 0000 and the error condition will not be set.

### 8278 Data Read

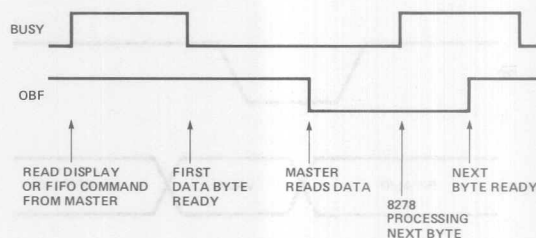
The master CPU can read DATA from the 8278 FIFO or Display buffers by using the A<sub>0</sub>,  $\overline{RD}$ , and D<sub>0</sub>-D<sub>7</sub> inputs as follows:



The master sends a  $\overline{RD}$  pulse with A<sub>0</sub>=0 and CS=0 and the 8278 responds by outputting data on lines D<sub>0</sub>-D<sub>7</sub>. The data is strobed by the trailing edge of  $\overline{RD}$ .

### Data Read Sequence

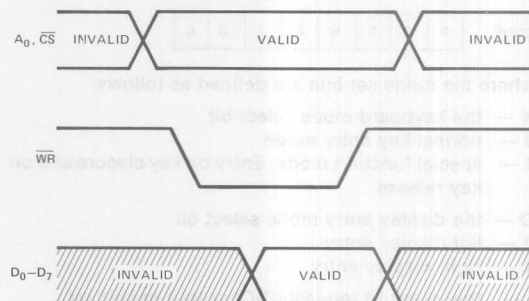
Before reading data, the master CPU must send a command to select FIFO or Display data. Following the command, the master must read STATUS and test the BUSY flag and the OBF flag to verify that the 8278 has responded to the previous command. A typical DATA READ sequence is as follows:



After the first read following a Read Display or Read FIFO command, successive reads may occur as soon as OBF rises.

### 8278 Data Write

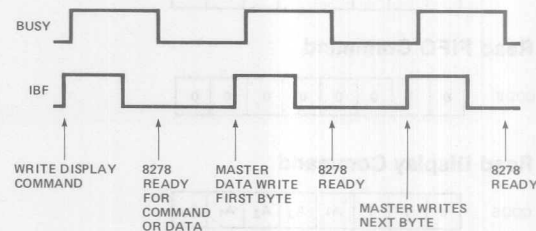
The master CPU can write DATA to the 8278 Display buffers by using the A<sub>0</sub>,  $\overline{WR}$  and D<sub>0</sub>-D<sub>7</sub> inputs as follows:



The master CPU presents the Data on the D<sub>0</sub>-D<sub>7</sub> lines with A<sub>0</sub>=0 and then sends a  $\overline{WR}$  pulse. The data is latched by the 8278 on the rising edge of  $\overline{WR}$ .

### Data Write Sequence

Before writing data to the 8278, the master CPU must first send a command to select the desired display entry mode and to specify the address of the next data byte. Following the commands, the master must read STATUS and test the BUSY flag (B) and IBF flag to verify that the 8278 has responded. A typical sequence is shown below:



## INTERFACE CONSIDERATIONS

### Scanned Keyboard Mode

With N-key rollover each key depression is treated independently from all others. When a key is depressed the debounce logic waits for a full scan of 128 keys and then checks to see if the key is still down. If it is, the key is entered into the FIFO.

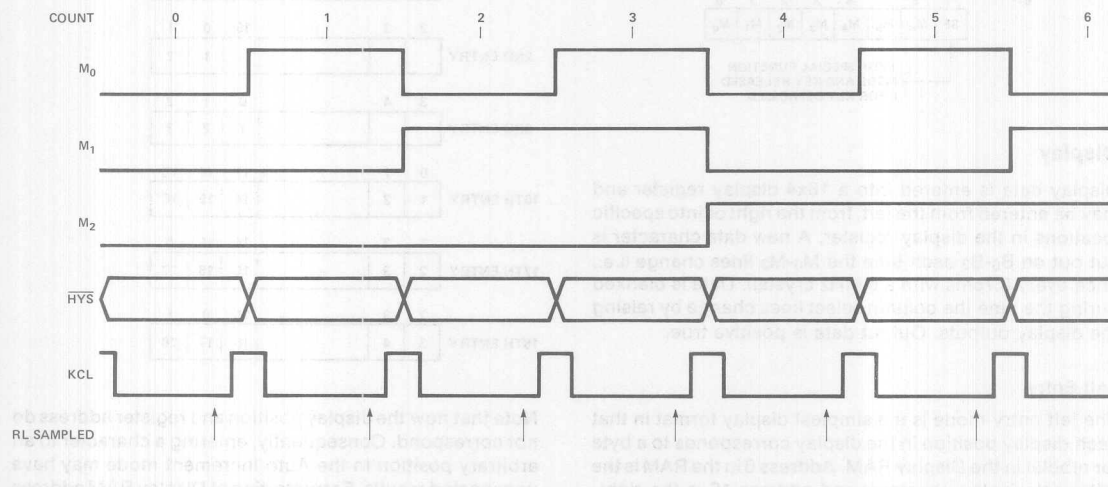


Figure 3. Keyboard Timing

If two key closures occur during the same scan the ERROR output is set, the KE flag is set in the Status word, the TONE output is activated and IRQ is set, and no further inputs are accepted. This condition is cleared by a high signal on the CLEAR input or by a system RESET input or by the CLEAR ERROR command.

In the special function mode both the key closure and the key release cause an entry to the FIFO. The release is entered with the MSB=1.

Any key entry triggers the TONE output for 10ms.

The HYS and KCL outputs enable the analog multiplexer and detector to be synchronized for interface to capacitive coupled keyboards.

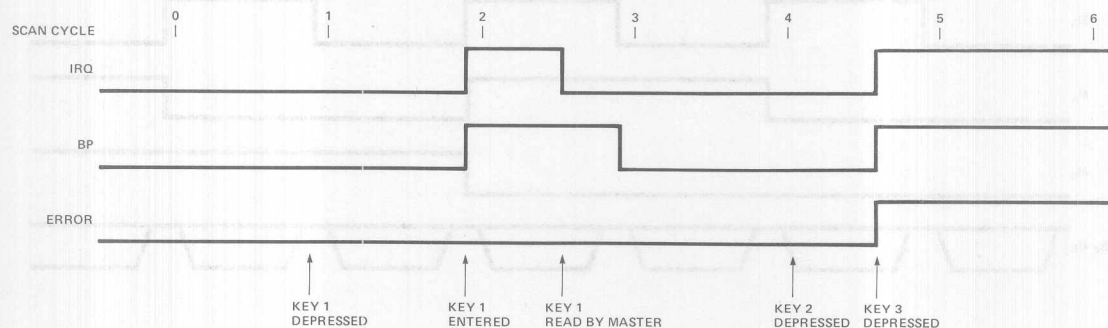
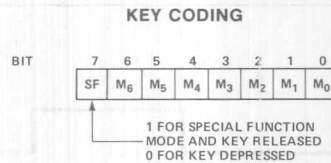


Figure 4. Key Entry and Error Timing

in the keyboard. The MSB is relevant only for special function keys in which code "0" signifies closure and "1" signifies release. The next four bits are the column count which indicates which column the key was found in. The last three bits are from the row counter.



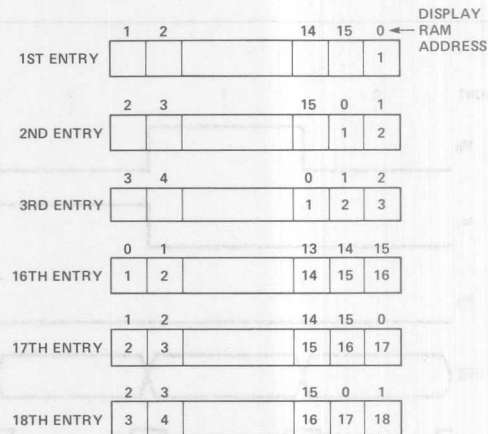
### Display

Display data is entered into a 16x4 display register and may be entered from the left, from the right or into specific locations in the display register. A new data character is put out on B<sub>0</sub>-B<sub>3</sub> each time the M<sub>6</sub>-M<sub>3</sub> lines change (i.e., once every 0.75ms with a 6 MHz crystal). Data is blanked during the time the column select lines change by raising the display outputs. Output data is positive true.

### Left Entry

The left entry mode is the simplest display format in that each display position in the display corresponds to a byte (or nibble) in the Display RAM. Address 0 in the RAM is the left-most display character and address 15 is the right-most display character. Entering characters from position zero causes the display to fill from the left. The 17th character is entered back in the left-most position and filling again proceeds from there.

character. The first entry is placed in the right-most display character. The next entry is also placed in the right-most character after the display is shifted left one character. The left-most character is shifted off the end and is lost.



Note that now the display position and register address do not correspond. Consequently, entering a character to an arbitrary position in the Auto Increment mode may have unexpected results. Entry starting at Display RAM address 0 with sequential entry is recommended. A Clear Display command should be given before display data is entered if the number of data characters is not equal to 16 (or 8) in this mode.

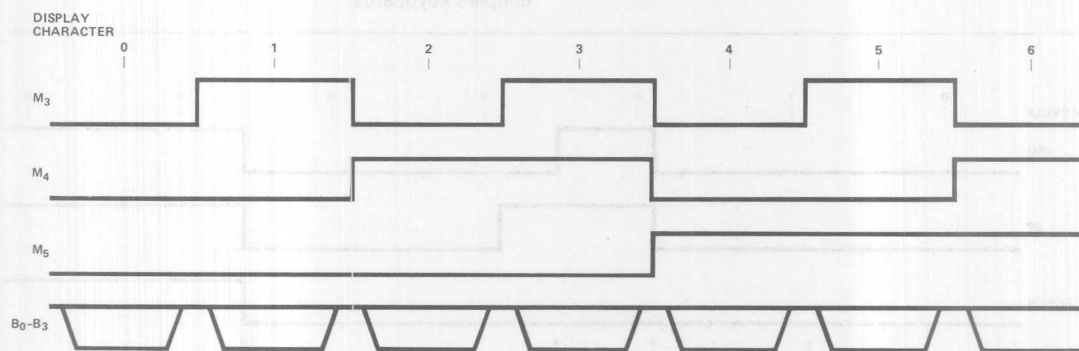
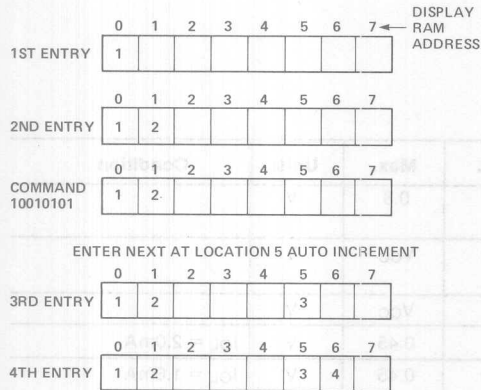


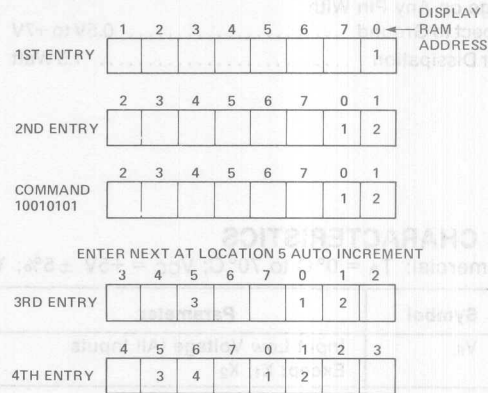
Figure 5. Display Timing

**Auto Increment**

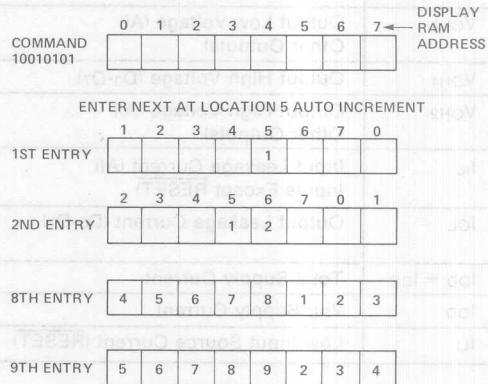
In the Left Entry mode, Auto Incrementing causes the address where the CPU will next write to be incremented by one and the character appears in the next location. With non-Auto Incrementing the entry is both to the same RAM address and display position. Entry to an arbitrary address in the Left Entry — Auto Increment mode has no undesirable side effects and the result is predictable:



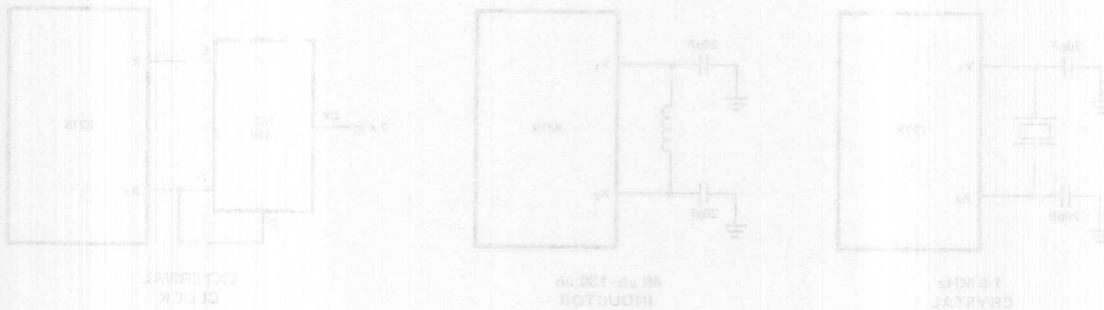
In the Right Entry mode, Auto Incrementing and non Incrementing have the same effect as in the Left Entry except that the address sequence is interrupted:



Starting at an arbitrary location operates as shown below:



Entry appears to be from the initial entry point.



**ABSOLUTE MAXIMUM RATINGS\***

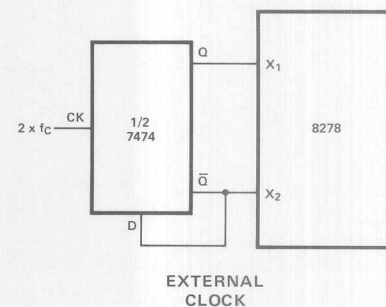
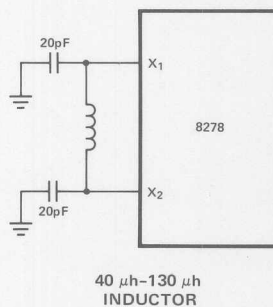
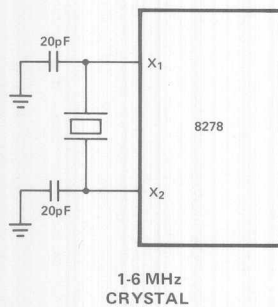
Ambient Temperature Under Bias ..... 0°C to 70°C  
 Storage Temperature ..... -65°C to +150°C  
 Voltage on Any Pin With  
 Respect to Ground ..... 0.5V to +7V  
 Power Dissipation ..... 1.5 Watt

\*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

**D.C. CHARACTERISTICS**

Commercial:  $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ;  $V_{CC} = +5\text{V} \pm 5\%$ ;  $V_{SS} = 0\text{V}$

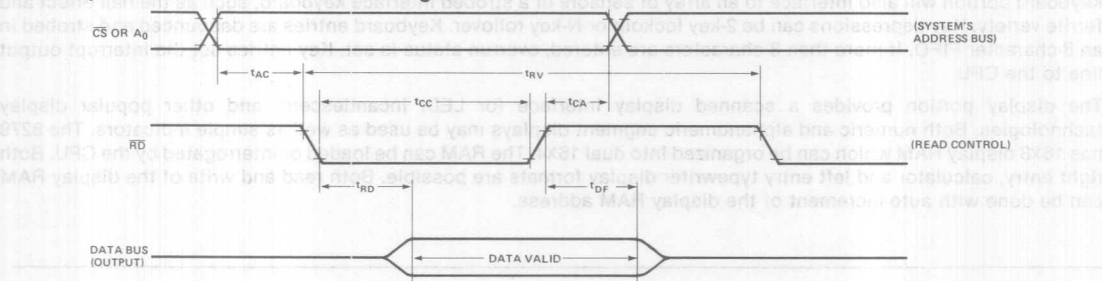
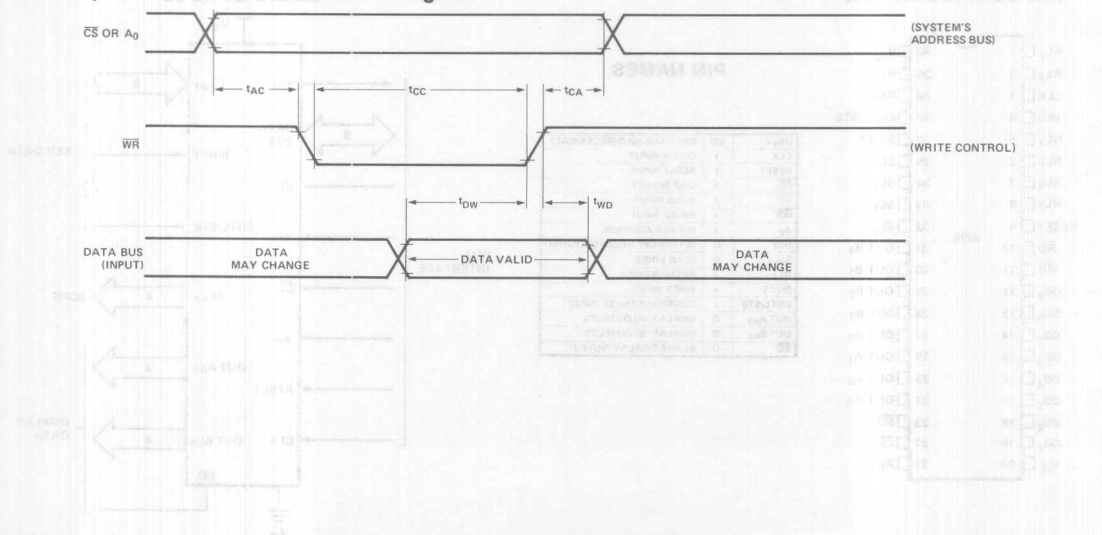
Symbol	Parameter	Min.	Max.	Units	Condition
$V_{IL}$	Input Low Voltage (All Inputs Except $X_1$ , $X_2$ )	-0.5	0.8	V	
$V_{IH1}$	Input High Voltage (All Inputs Except $X_1$ , $X_2$ , RESET)	2.0	$V_{CC}$	V	
$V_{IH2}$	RESET High Voltage	3.0	$V_{CC}$	V	
$V_{OL1}$	Output Low Voltage ( $D_0$ - $D_7$ )		0.45	V	$I_{OL} = 2.0\text{mA}$
$V_{OL2}$	Output Low Voltage (All Other Outputs)		0.45	V	$I_{OL} = 1.6\text{mA}$
$V_{OH1}$	Output High Voltage ( $D_0$ - $D_7$ )	2.4		V	$I_{OH} = -400\mu\text{A}$
$V_{OH2}$	Output High Voltage (All Other Outputs)	2.4		V	$I_{OH} = -50\mu\text{A}$
$I_{IL}$	Input Leakage Current (All Inputs Except RESET)		$\pm 10$	$\mu\text{A}$	$V_{IN} = V_{CC}$
$I_{OL}$	Output Leakage Current ( $D_0$ - $D_7$ )		$\pm 10$	$\mu\text{A}$	$V_{IN} = V_{SS} + 0.45\text{V}$ or $V_{IN} = V_{CC}$
$I_{DD} + I_{CC}$	Total Supply Current		135	mA	$V_{CC} = 5.5\text{V}$
$I_{DD}$	$V_{DD}$ Supply Current		25	mA	$V_{CC} = 5.5\text{V}$
$I_{LI}$	Low Input Source Current (RESET)		0.2	mA	$V_{IL} = 0.8\text{V}$

**8278 CLOCK OPTIONS**

**A.C. CHARACTERISTICS**

$T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ;  $V_{CC} = +5\text{V} \pm 10\%$ ;  $V_{SS} = 0\text{V}$

Symbol	Parameter	Min.	Max.	Units	Condition
$t_{AC}$	Address ( $\overline{CS}$ , $A_0$ ) Setup to Control ( $\overline{RD}$ , $\overline{WR}$ )	0		ns	$D_0$ - $D_7$ , $C_L = 150\text{pF}$
$t_{CA}$	Address Hold from Control	0		ns	
$t_{CC}$	Control Pulse Width	250		ns	
$t_{DW}$	Data in Setup to $\overline{WR}$ T.E.	150		ns	
$t_{WD}$	Data in Hold After $\overline{WR}$ T.E.	0		ns	
$t_{RD}$	$\overline{RD}$ L.E. to Data Out Valid		150	ns	
$t_{DF}$	$\overline{RD}$ T.E. to Data Out Float	10	100	ns	With 6MHz Crystal
$t_{MCY}$	Matrix Cycle Time		10.7	ms	
$t_{RV}$	Recovery Time Between Reads and/or Writes	1		$\mu\text{s}$	

**WAVEFORMS****Read Operation — Data Bus Buffer Register****Write Operation — Data Bus Buffer Register**

# 8279/8279-5

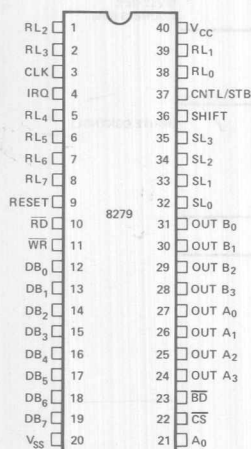
## PROGRAMMABLE KEYBOARD/DISPLAY INTERFACE

- MCS-85™ Compatible 8279-5
- Simultaneous Keyboard Display Operations
- Scanned Keyboard Mode
- Scanned Sensor Mode
- Strobed Input Entry Mode
- 8-Character Keyboard FIFO
- 2-Key Lockout or N-Key Rollover with Contact Debounce
- Dual 8- or 16-Numerical Display
- Single 16-Character Display
- Right or Left Entry 16-Byte Display RAM
- Mode Programmable from CPU
- Programmable Scan Timing
- Interrupt Output on Key Entry

The Intel® 8279 is a general purpose programmable keyboard and display I/O interface device designed for use with Intel® microprocessors. The keyboard portion can provide a scanned interface to a 64-contact key matrix. The keyboard portion will also interface to an array of sensors or a strobed interface keyboard, such as the hall effect and ferrite variety. Key depressions can be 2-key lockout or N-key rollover. Keyboard entries are debounced and strobed in an 8-character FIFO. If more than 8 characters are entered, overrun status is set. Key entries set the interrupt output line to the CPU.

The display portion provides a scanned display interface for LED, incandescent, and other popular display technologies. Both numeric and alphanumeric segment displays may be used as well as simple indicators. The 8279 has 16X8 display RAM which can be organized into dual 16X4. The RAM can be loaded or interrogated by the CPU. Both right entry, calculator and left entry typewriter display formats are possible. Both read and write of the display RAM can be done with auto-increment of the display RAM address.

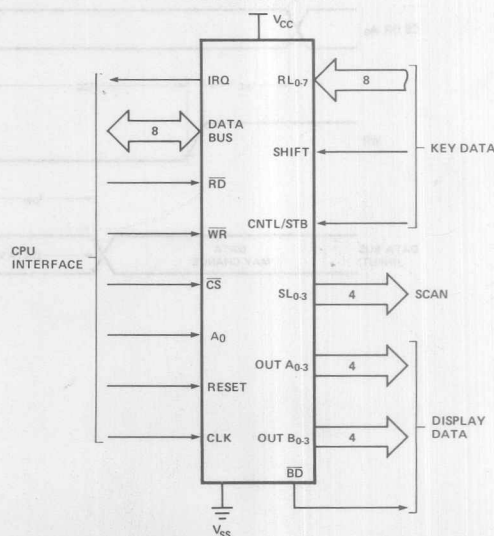
### PIN CONFIGURATION



### PIN NAMES

Pin	I/O	Function
DB0-7	I/O	DATA BUS (BI-DIRECTIONAL)
CLK	I	CLOCK INPUT
RESET	I	RESET INPUT
CS	I	CHIP SELECT
RD	I	READ INPUT
WR	I	WRITE INPUT
A0	I	BUFFER ADDRESS
IRQ	O	INTERRUPT REQUEST OUTPUT
SL0-3	O	SCAN LINES
RL0-7	I	RETURN LINES
SHIFT	I	SHIFT INPUT
CNTL/STB	I	CONTROL/STROBE INPUT
OUT A0-3	O	DISPLAY (A) OUTPUTS
OUT B0-3	O	DISPLAY (B) OUTPUTS
BD	O	BLANK DISPLAY OUTPUT

### LOGIC SYMBOL



## FUNCTIONAL DESCRIPTION

Since data input and display are an integral part of many microprocessor designs, the system designer needs an interface that can control these functions without placing a large load on the CPU. The 8279 provides this function for 8-bit microprocessors.

The 8279 has two sections: keyboard and display. The keyboard section can interface to regular typewriter style keyboards or random toggle or thumb switches. The display section drives alphanumeric displays or a bank of indicator lights. Thus the CPU is relieved from scanning the keyboard or refreshing the display.

The 8279 is designed to directly connect to the microprocessor bus. The CPU can program all operating modes for the 8279. These modes include:

### Input Modes

- Scanned Keyboard — with encoded (8 x 8 x 4 key keyboard) or decoded (4 x 8 x 4 key keyboard) scan lines. A key depression generates a 6-bit encoding of key position. Position and shift and control status are stored in the FIFO. Keys are automatically debounced with 2-key lockout or N-key rollover.

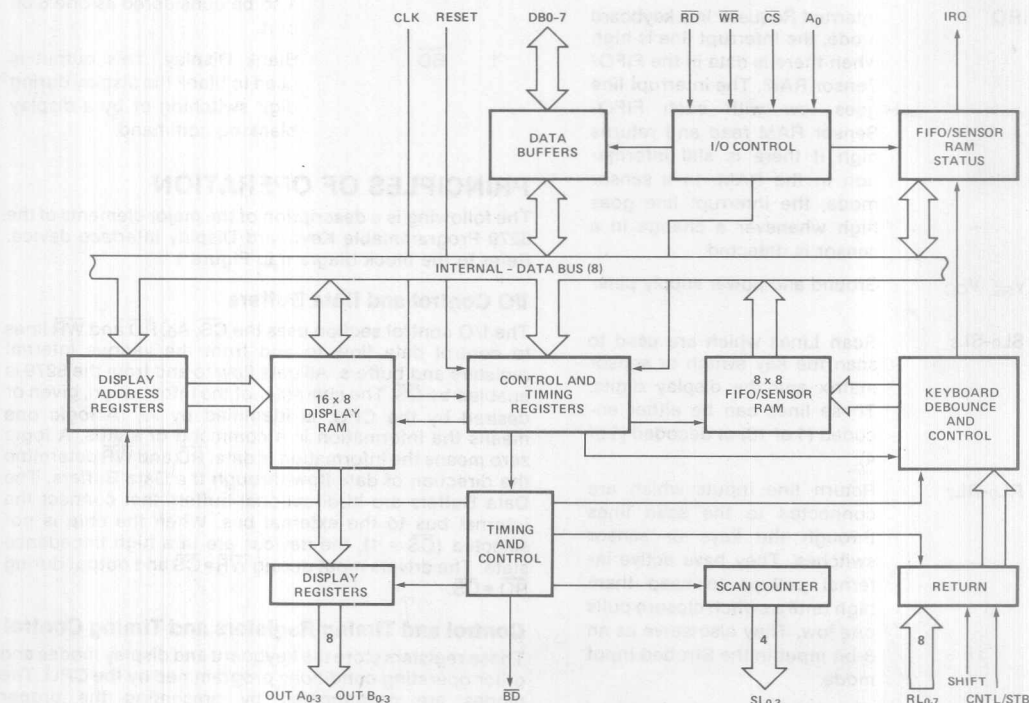
- Scanned Sensor Matrix — with encoded (8 x 8 matrix switches) or decoded (4 x 8 matrix switches) scan lines. Key status (open or closed) stored in RAM addressable by CPU.
- Strobed Input — Data on return lines during control line strobe is transferred to FIFO.

### Output Modes

- 8 or 16 character multiplexed displays that can be organized as dual 4-bit or single 8-bit.
- Right entry or left entry display formats.

Other features of the 8279 include:

- Mode programming from the CPU.
- Programmable clock to match the 8279 scan times to the CPU cycle time.
- Interrupt output to signal CPU when there is keyboard or sensor data available.
- An 8 byte FIFO to store keyboard information.
- 16 byte internal Display RAM for display refresh. This RAM can also be read by the CPU.



## HARDWARE DESCRIPTION

The 8279 is packaged in a 40 pin DIP. The following is a functional description of each pin.

No. Of Pins	Designation	Function
8	DB <sub>0</sub> -DB <sub>7</sub>	Bi-directional data bus. All data and commands between the CPU and the 8279 are transmitted on these lines.
1	CLK	Clock from system used to generate internal timing.
1	RESET	A high signal on this pin resets the 8279.
1	$\overline{CS}$	Chip Select. A low on this pin enables the interface functions to receive or transmit.
1	A <sub>0</sub>	Buffer Address. A high on this line indicates the signals in or out are interpreted as a command or status. A low indicates that they are data.
2	$\overline{RD}$ , $\overline{WR}$	Input/Output read and write. These signals enable the data buffers to either send data to the external bus or receive it from the external bus.
1	IRQ	Interrupt Request. In a keyboard mode, the interrupt line is high when there is data in the FIFO/Sensor RAM. The interrupt line goes low with each FIFO/Sensor RAM read and returns high if there is still information in the RAM. In a sensor mode, the interrupt line goes high whenever a change in a sensor is detected.
2	V <sub>SS</sub> , V <sub>CC</sub>	Ground and power supply pins.
4	SL <sub>0</sub> -SL <sub>3</sub>	Scan Lines which are used to scan the key switch or sensor matrix and the display digits. These lines can be either encoded (1 of 16) or decoded (1 of 4).
8	RL <sub>0</sub> -RL <sub>7</sub>	Return line inputs which are connected to the scan lines through the keys or sensor switches. They have active internal pullups to keep them high until a switch closure pulls one low. They also serve as an 8-bit input in the Strobed Input mode.
1	SHIFT	The shift input status is stored along with the key position on key closure in the Scanned

No. Of Pins	Designation	Function
1	CNTL/STB	Keyboard modes. It has an active internal pullup to keep it high until a switch closure pulls it low. For keyboard modes this line is used as a control input and stored like status on a key closure. The line is also the strobe line that enters the data into the FIFO in the Strobed Input mode. (Rising Edge). It has an active internal pullup to keep it high until a switch closure pulls it low.
4	OUT A <sub>0</sub> -OUT A <sub>3</sub>	These two ports are the outputs for the 16 x 4 display refresh registers. The data from these outputs is synchronized to the scan lines (SL <sub>0</sub> -SL <sub>3</sub> ) for multiplexed digit displays. The two 4 bit ports may be blanked independently. These two ports may also be considered as one 8 bit port.
4	OUT B <sub>0</sub> -OUT B <sub>3</sub>	
1	$\overline{BD}$	Blank Display. This output is used to blank the display during digit switching or by a display blanking command.

## PRINCIPLES OF OPERATION

The following is a description of the major elements of the 8279 Programmable Keyboard/Display interface device. Refer to the block diagram in Figure 1.

### I/O Control and Data Buffers

The I/O control section uses the  $\overline{CS}$ , A<sub>0</sub>,  $\overline{RD}$  and  $\overline{WR}$  lines to control data flow to and from the various internal registers and buffers. All data flow to and from the 8279 is enabled by  $\overline{CS}$ . The character of the information, given or desired by the CPU, is identified by A<sub>0</sub>. A logic one means the information is a command or status. A logic zero means the information is data.  $\overline{RD}$  and  $\overline{WR}$  determine the direction of data flow through the Data Buffers. The Data Buffers are bi-directional buffers that connect the internal bus to the external bus. When the chip is not selected ( $\overline{CS} = 1$ ), the devices are in a high impedance state. The drivers input during  $\overline{WR} \bullet \overline{CS}$  and output during  $\overline{RD} \bullet \overline{CS}$ .

### Control and Timing Registers and Timing Control

These registers store the keyboard and display modes and other operating conditions programmed by the CPU. The modes are programmed by presenting the proper command on the data lines with A<sub>0</sub> = 1 and then sending a  $\overline{WR}$ . The command is latched on the rising edge of  $\overline{WR}$ .

The command is then decoded and the appropriate function is set. The timing control contains the basic timing counter chain. The first counter is a  $\div N$  prescaler that can be programmed to match the CPU cycle time to the internal timing. The prescaler is software programmed to a value between 2 and 31. A value which yields an internal frequency of 100 kHz gives a 5.1 ms keyboard scan time and a 10.3 ms debounce time. The other counters divide down the basic internal frequency to provide the proper key scan, row scan, keyboard matrix scan, and display scan times.

### Scan Counter

The scan counter has two modes. In the encoded mode, the counter provides a binary count that must be externally decoded to provide the scan lines for the keyboard and display. In the decoded mode, the scan counter decodes the least significant 2 bits and provides a decoded 1 of 4 scan. Note that when the keyboard is in decoded scan, so is the display. This means that only the first 4 characters in the Display RAM are displayed.

In the encoded mode, the scan lines are active high outputs. In the decoded mode, the scan lines are active low outputs.

### Return Buffers and Keyboard Debounce and Control

The 8 return lines are buffered and latched by the Return Buffers. In the keyboard mode, these lines are scanned, looking for key closures in that row. If the debounce circuit detects a closed switch, it waits about 10 msec to check if the switch remains closed. If it does, the address of the switch in the matrix plus the status of SHIFT and CONTROL are transferred to the FIFO. In the scanned Sensor Matrix modes, the contents of the return lines is directly transferred to the corresponding row of the Sensor RAM (FIFO) each key scan time. In Strobed Input mode, the contents of the return lines are transferred to the FIFO on the rising edge of the CNTL/STB line pulse.

### FIFO/Sensor RAM and Status

This block is a dual function 8 x 8 RAM. In Keyboard or Strobed Input modes, it is a FIFO. Each new entry is written into successive RAM positions and each is then read in order of entry. FIFO status keeps track of the number of characters in the FIFO and whether it is full or empty. Too many reads or writes will be recognized as an error. The status can be read by an RD with CS low and A<sub>0</sub> high. The status logic also provides an IRQ signal when the FIFO is not empty. In Scanned Sensor Matrix mode, the memory is a Sensor RAM. Each row of the Sensor RAM is loaded with the status of the corresponding row of sensor in the sensor matrix. In this mode, IRQ is high if a change in a sensor is detected.

### Display Address Registers and Display RAM

The Display Address Registers hold the address of the word currently being written or read by the CPU and the two 4-bit nibbles being displayed. The read/write addresses are programmed by CPU command. They also can be set to auto increment after each read or write. The Display RAM can be directly read by the CPU after the correct mode and address is set. The addresses for the A and B nibbles are automatically updated by the 8279 to match data entry by the CPU. The A and B nibbles can be entered independently or as one word, according to the mode that is set by the CPU. Data entry to the display can be set to either left or right entry. See Interface Considerations for details.

## SOFTWARE OPERATION

### 8279 commands

The following commands program the 8279 operating modes. The commands are sent on the Data Bus with CS low and A<sub>0</sub> high and are loaded to the 8279 on the rising edge of WR.

### Keyboard/Display Mode Set

Code: 

MSB								LSB	
0	0	0	D	D	K	K	K	K	

Where DD is the Display Mode and KKK is the Keyboard Mode.

### DD

- 0 0 8 8-bit character display — Left entry
- 0 1 16 8-bit character display — Left entry\*
- 1 0 8 8-bit character display — Right entry
- 1 1 16 8-bit character display — Right entry

For description of right and left entry, see Interface Considerations. Note that when decoded scan is set in keyboard mode, the display is reduced to 4 characters independent of display mode set.

### KKK

- 0 0 0 Encoded Scan Keyboard — 2 Key Lockout
- 0 0 1 Decoded Scan Keyboard — 2-Key Lockout
- 0 1 0 Encoded Scan Keyboard — N-Key Rollover
- 0 1 1 Decoded Scan Keyboard — N-Key Rollover
- 1 0 0 Encoded Scan Sensor Matrix
- 1 0 1 Decoded Scan Sensor Matrix
- 1 1 0 Strobed Input, Encoded Display Scan
- 1 1 1 Strobed Input, Decoded Display Scan

### Program Clock

Code: 

0	0	1	P	P	P	P	P
---	---	---	---	---	---	---	---

Where P P P P P is the prescaler value 2 to 31. The programmable prescaler divides the external clock by P P P P P to get the basic internal frequency. Choosing a divisor that yields 100 KHz will give the specified scan and debounce times. Default after a reset pulse (but not a program clear) is 31.

### Read FIFO/Sensor RAM

Code: 

0	1	0	AI	X	A	A	A
---	---	---	----	---	---	---	---

 X = Don't Care

Where AI is the Auto-Increment flag for the Sensor RAM and AAA is the row that is going to be read by the CPU. AI and AAA are used only if the mode is set to Sensor Matrix. This command is used to specify that the source of data reads ( $\overline{CS} \cdot RD \cdot \overline{A_0}$ ) by the CPU is the FIFO/Sensor RAM. No additional commands are necessary as long as

\*Default after reset.

data is desired from the FIFO/Sensor RAM. Another command is necessary if reading is desired from a different row than has been selected. If AI is a one, the row select counter will be incremented after each read so the next read will be from the next Sensor RAM row.

In the Auto Increment mode for reading data from the FIFO/Sensor RAM, each read advances the address by one so that the next read is from the next character. This Auto Incrementing has no effect on the display.

### Read Display RAM

Code: 

0	1	1	AI	A	A	A	A
---	---	---	----	---	---	---	---

Where AI is the Auto-Increment flag for the Display RAM and AAAA is the character that the CPU is going to read next. Since the CPU uses the same counter for reading and writing, this command also sets the next write location and Auto-Increment mode. This command is used to specify the display RAM as the data source for CPU data reads. If AI is set, the character address will be incremented after each read (or write) so that the next read (or write) will be from (to) the next character.

### Write Display RAM

Code: 

1	0	0	AI	A	A	A	A
---	---	---	----	---	---	---	---

Where AI is the Auto-Increment flag for the Display RAM and AAAA is the character that the CPU is going to write next. The addressing and Auto-Increment are identical to Read Display RAM. The difference is that Write Display RAM does not affect the source of CPU reads. The CPU will read from whichever RAM (Display or FIFO/Sensor) was last specified. This command will, however, change the location the next Display RAM read will be from if that source was specified.

### Display Write Inhibit/Blanking

Code: 

1	0	1	X	IW	IW	BL	BL
				A	B	A	B

Where IW is Inhibit Writing (nibble A or B) and BL is Blanking (nibble A or B). If the display is being used as a dual 4-bit display, then it is necessary to mask one of the 4-bit halves so that entries to the Display from the CPU do not affect the other half. The IW flags allow the programmer to do this. It is also useful to be able to blank either half when that half is not to be displayed. The BL flags blank the display. The next command sets the output code to be used as a "blank". Default after reset is all zeros. Note that to blank a display formatted as a single 8-bit output, it is necessary to set both BL flags to entirely blank the display. A "1" sets the flag. Reissuing the command with a "0" resets the flag.

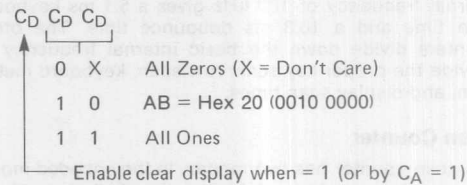
### Clear

Code: 

1	1	0	C <sub>D</sub>	C <sub>D</sub>	C <sub>D</sub>	C <sub>F</sub>	C <sub>A</sub>
---	---	---	----------------	----------------	----------------	----------------	----------------

Where C<sub>D</sub> is Clear Display, C<sub>F</sub> is Clear FIFO Status (including interrupt), and C<sub>A</sub> is Clear All. C<sub>D</sub> is used to

clear all positions of the Display RAM to a programmable code. All ones, all zeros and hexadecimal 20 are possible. The 2 least significant bits of C<sub>D</sub> are also used to specify the blanking code (see below).



Clearing the display takes approximately 160  $\mu$ s. During this time the CPU cannot write to the Display RAM. The MSB of the FIFO status word will be set during this time. C<sub>F</sub> set the FIFO status to empty and resets the interrupt output line. After execution of a clear command with C<sub>F</sub> set, the Sensor Matrix mode RAM pointer will be set to row 0.

C<sub>A</sub> has the combined effect of C<sub>D</sub> and C<sub>F</sub>. C<sub>A</sub> uses the C<sub>D</sub> clearing code to determine how to clear the Display RAM. C<sub>A</sub> also resets the internal timing chain to resynchronize it.

### End Interrupt/Error Mode Set

Code: 

1	1	1	E	X	X	X	X
---	---	---	---	---	---	---	---

 X = Don't care.

For the sensor matrix modes this command lowers the IRQ line and enables further writing into RAM. (The IRQ line would have been raised upon the detection of a change in a sensor value. This would have also inhibited further writing into the RAM until reset).

For the N-key rollover mode — if the E bit is programmed to "1" the chip will operate in the special Error mode. (For further details, see Interface Considerations Section.)

### Status Word

The status word contains the FIFO status, error, and display unavailable signals. This word is read by the CPU when A<sub>0</sub> is high and  $\overline{CS}$  and  $\overline{RD}$  are low. See Interface Considerations for more detail on status word.

### Data Read

Data is read when A<sub>0</sub>,  $\overline{CS}$  and  $\overline{RD}$  are all low. The source of the data is specified by the Read FIFO or Read Display commands. The trailing edge of  $\overline{RD}$  will cause the address of the RAM being read to be incremented if the Auto-Increment flag is set. FIFO reads always increment (if no error occurs) independent of AI.

### Data Write

Data that is written with A<sub>0</sub>,  $\overline{CS}$  and  $\overline{WR}$  low is always written to the Display RAM. The address is specified by the latest Read Display or Write Display command. Auto-Incrementing on the rising edge of  $\overline{WR}$  occurs if AI set by the latest display command.

## INTERFACE CONSIDERATIONS

### Scanned Keyboard Mode, 2-Key Lockout

There are three possible combinations of conditions that can occur during debounce scanning. When a key is depressed, the debounce logic is set. Other depressed keys are looked for during the next two scans. If none are encountered, it is a single key depression and the key position is entered into the FIFO along with the status of CNTL and SHIFT lines. If the FIFO was empty, IRQ will be set to signal the CPU that there is an entry in the FIFO. If the FIFO was full, the key will not be entered and the error flag will be set. If another closed switch is encountered, no entry to the FIFO can occur. If all other keys are released before this one, then it will be entered to the FIFO. If this key is released before any other, it will be entirely ignored. A key is entered to the FIFO only once per depression, no matter how many keys were pressed along with it or in what order they were released. If two keys are depressed within the debounce cycle, it is a simultaneous depression. Neither key will be recognized until one key remains depressed alone. The last key will be treated as a single key depression.

### Scanned Keyboard Mode, N-Key Rollover

With N-key Rollover each key depression is treated independently from all others. When a key is depressed, the debounce circuit waits 2 keyboard scans and then checks to see if the key is still down. If it is, the key is entered into the FIFO. Any number of keys can be depressed and another can be recognized and entered into the FIFO. If a simultaneous depression occurs, the keys are recognized and entered according to the order the keyboard scan found them.

### Scanned Keyboard — Special Error Modes

For N-key rollover mode the user can program a special error mode. This is done by the "End Interrupt/Error Mode Set" command. The debounce cycle and key-validity check are as in normal N-key mode. If during a single debounce cycle, two keys are found depressed, this is considered a simultaneous multiple depression, and sets an error flag. This flag will prevent any further writing into the FIFO and will set interrupt (if not yet set). The error flag could be read in this mode by reading the FIFO STATUS word. (See "FIFO STATUS" for further details.) The error flag is reset by sending the normal CLEAR command with CF = 1.

### Sensor Matrix Mode

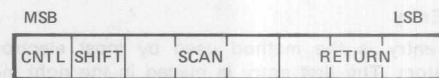
In Sensor Matrix mode, the debounce logic is inhibited. The status of the sensor switch is inputted directly to the Sensor RAM. In this way the Sensor RAM keeps an image of the state of the switches in the sensor matrix. Although debouncing is not provided, this mode has the advantage that the CPU knows how long the sensor was closed and when it was released. A keyboard mode can only indicate a validated closure. To make the software easier, the designer should functionally group the sensors by row since this is the format in which the CPU will read them. The IRQ line goes high if any sensor value change is detected at the end of a sensor matrix scan. The IRQ line is cleared by the first data read operation if the Auto-

Increment flag is set to zero, or by the End Interrupt command if the Auto-Increment flag is set to one.

**Note:** Multiple changes in the matrix Addressed by (SL<sub>0</sub>-3 = 0) may cause multiple interrupts. (SL<sub>0</sub> = 0 in the Decoded Mode). Reset may cause the 8279 to see multiple changes.

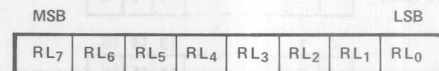
### Data Format

In the Scanned Keyboard mode, the character entered into the FIFO corresponds to the position of the switch in the keyboard plus the status of the CNTL and SHIFT lines (non-inverted). CNTL is the MSB of the character and SHIFT is the next most significant bit. The next three bits are from the scan counter and indicate the row the key was found in. The last three bits are from the column counter and indicate to which return line the key was connected.

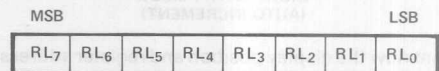


SCANNED KEYBOARD DATA FORMAT

In Sensor Matrix mode, the data on the return lines is entered directly in the row of the Sensor RAM that corresponds to the row in the matrix being scanned. Therefore, each switch position maps directly to a Sensor RAM position. The SHIFT and CNTL inputs are ignored in this mode. Note that switches are not necessarily the only thing that can be connected to the return lines in this mode. Any logic that can be triggered by the scan lines can enter data to the return line inputs. Eight multiplexed input ports could be tied to the return lines and scanned by the 8279.



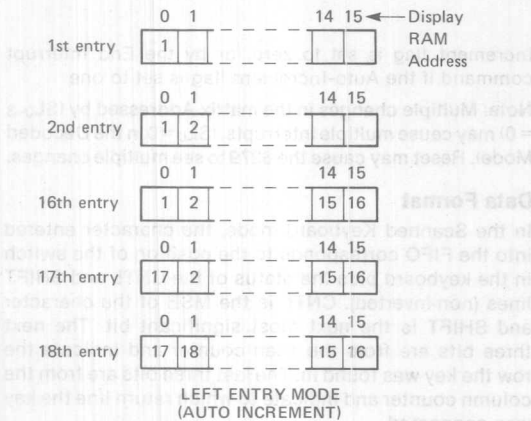
In Strobed Input mode, the data is also entered to the FIFO from the return lines. The data is entered by the rising edge of a CNTL/STB line pulse. Data can come from another encoded keyboard or simple switch matrix. The return lines can also be used as a general purpose strobed input.



### Display

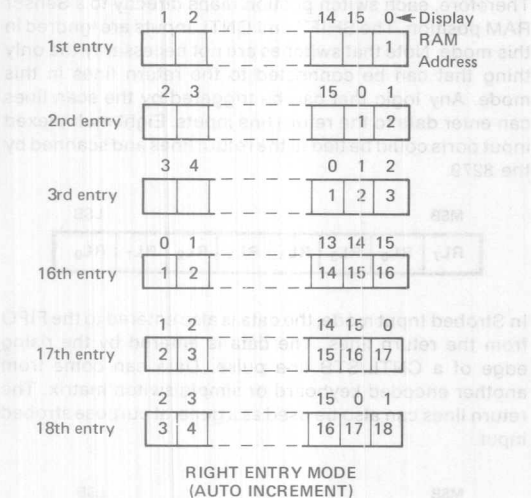
#### Left Entry

Left Entry mode is the simplest display format in that each display position directly corresponds to a byte (or nibble) in the Display RAM. Address 0 in the RAM is the left-most display character and address 15 (or address 7 in 8 character display) is the right most display character. Entering characters from position zero causes the display to fill from the left. The 17th (9th) character is entered back in the left most position and filling again proceeds from there.



### Right Entry

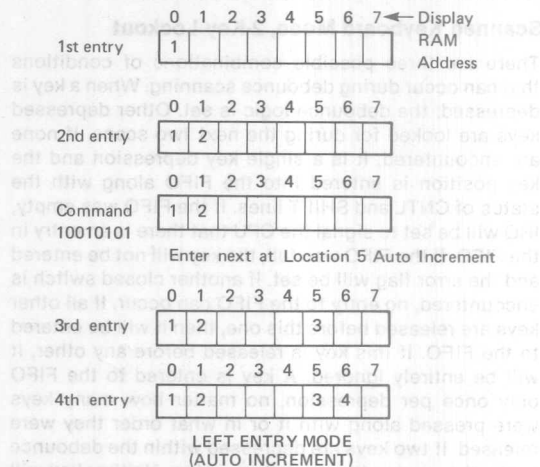
Right entry is the method used by most electronic calculators. The first entry is placed in the right most display character. The next entry is also placed in the right most character after the display is shifted left one character. The left most character is shifted off the end and is lost.



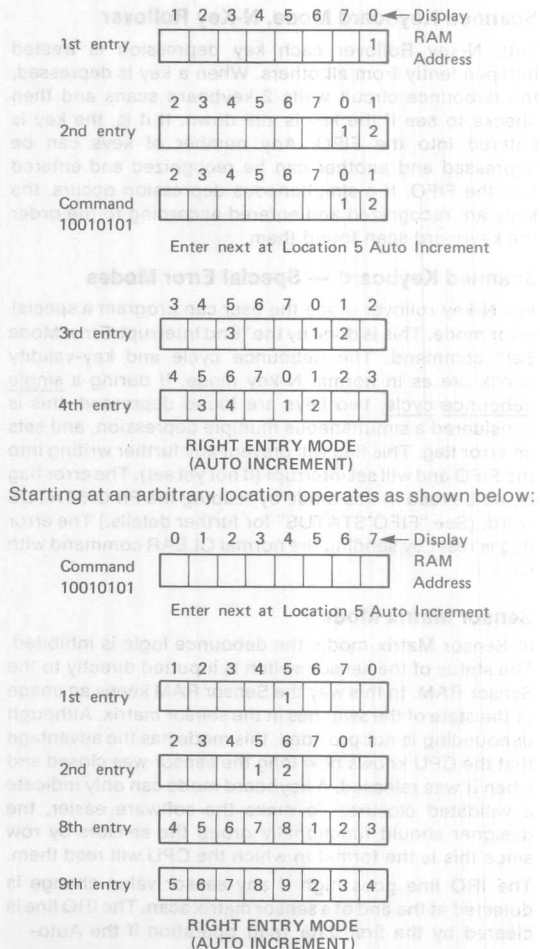
Note that now the display position and register address do not correspond. Consequently, entering a character to an arbitrary position in the Auto Increment mode may have unexpected results. Entry starting at Display RAM address 0 with sequential entry is recommended.

### Auto Increment

In the Left Entry mode, Auto Incrementing causes the address where the CPU will next write to be incremented by one and the character appears in the next location. With non-Auto Incrementing the entry is both to the same RAM address and display position. Entry to an arbitrary address in the Auto Increment mode has no undesirable side effects and the result is predictable:



In the Right Entry mode, Auto Incrementing and non Incrementing have the same effect as in the Left Entry except if the address sequence is interrupted:



Entry appears to be from the initial entry point.

### 8/16 Character Display Formats

If the display mode is set to an 8 character display, the on duty-cycle is double what it would be for a 16 character display (e.g., 5.1 ms scan time for 8 characters vs. 10.3 ms for 16 characters with 100 kHz internal frequency).

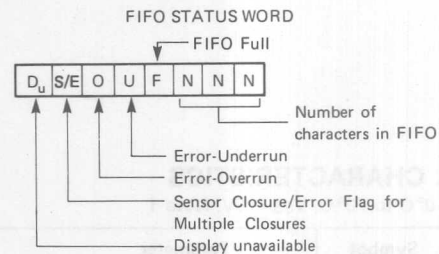
### G. FIFO Status

FIFO status is used in the Keyboard and Strobed Input modes to indicate the number of characters in the FIFO and to indicate whether an error has occurred. There are two types of errors possible: overrun and underrun. Overrun occurs when the entry of another character into a full FIFO is attempted. Underrun occurs when the CPU tries to read an empty FIFO.

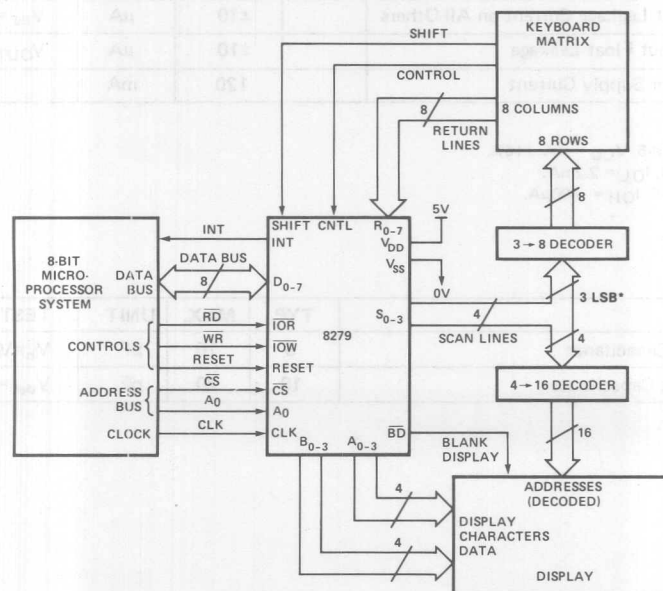
The FIFO status word also has a bit to indicate that the Display RAM was unavailable because a Clear Display or Clear All command had not completed its clearing operation.

In a Sensor Matrix mode, a bit is set in the FIFO status word to indicate that at least one sensor closure indication is contained in the Sensor RAM.

In Special Error Mode the S/E bit is showing the error flag and serves as an indication to whether a simultaneous multiple closure error has occurred.



## APPLICATIONS



\*Do not drive the keyboard decoder with the MSB of the scan lines.

**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature	0°C to 70°C
Storage Temperature	-65°C to 125°C
Voltage on any Pin with Respect to Ground	-0.5V to +7V
Power Dissipation	1 Watt

\*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

**D.C. CHARACTERISTICS**

T<sub>A</sub> = 0°C to 70°C, V<sub>SS</sub> = 0V, Note 1

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V <sub>IL1</sub>	Input Low Voltage for Return Lines	-0.5	1.4	V	
V <sub>IL2</sub>	Input Low Voltage for All Others	-0.5	0.8	V	
V <sub>IH1</sub>	Input High Voltage for Return Lines	2.2		V	
V <sub>IH2</sub>	Input High Voltage for All Others	2.0		V	
V <sub>OL</sub>	Output Low Voltage		0.45	V	Note 2
V <sub>OH</sub>	Output High Voltage on Interrupt Line	3.5		V	Note 3
I <sub>IL1</sub>	Input Current on Shift, Control and Return Lines		+10 -100	μA	V <sub>IN</sub> = V <sub>CC</sub> V <sub>IN</sub> = 0V
I <sub>IL2</sub>	Input Leakage Current on All Others		±10	μA	V <sub>IN</sub> = V <sub>CC</sub> to 0V
I <sub>OFL</sub>	Output Float Leakage		±10	μA	V <sub>OUT</sub> = V <sub>CC</sub> to 0V
I <sub>CC</sub>	Power Supply Current		120	mA	

**Notes:**

1. 8279, V<sub>CC</sub> = +5V ±5%; 8279-5, V<sub>CC</sub> = +5V ±10%.
2. 8279, I<sub>OL</sub> = 1.6mA; 8279-5, I<sub>OL</sub> = 2.2mA.
3. 8279, I<sub>OH</sub> = -100μA; 8279-5, I<sub>OH</sub> = -400μA.

**CAPACITANCE**

SYMBOL	TEST	TYP.	MAX.	UNIT	TEST CONDITIONS
C <sub>in</sub>	Input Capacitance	5	10	pF	V <sub>in</sub> = V <sub>CC</sub>
C <sub>out</sub>	Output Capacitance	10	20	pF	V <sub>out</sub> = V <sub>CC</sub>

**A.C. CHARACTERISTICS** $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{SS} = 0\text{V}$ , (Note 1)**Bus Parameters****Read Cycle:**

Symbol	Parameter	8279		8279-5		Unit
		Min.	Max.	Min.	Max.	
$t_{AR}$	Address Stable Before $\overline{\text{READ}}$	50		0		ns
$t_{RA}$	Address Hold Time for $\overline{\text{READ}}$	5		0		ns
$t_{RR}$	$\overline{\text{READ}}$ Pulse Width	420		250		ns
$t_{RD}^{[2]}$	Data Delay from $\overline{\text{READ}}$		300		150	ns
$t_{AD}^{[2]}$	Address to Data Valid		450		250	ns
$t_{DF}$	$\overline{\text{READ}}$ to Data Floating	10	100	10	100	ns
$t_{RCY}$	Read Cycle Time	1		1		$\mu\text{s}$

**Write Cycle:**

Symbol	Parameter	8279		8279-5		Unit
		Min.	Max.	Min.	Max.	
$t_{AW}$	Address Stable Before $\overline{\text{WRITE}}$	50		0		ns
$t_{WA}$	Address Hold Time for $\overline{\text{WRITE}}$	20		0		ns
$t_{WW}$	$\overline{\text{WRITE}}$ Pulse Width	400		250		ns
$t_{DW}$	Data Set Up Time for $\overline{\text{WRITE}}$	300		150		ns
$t_{WD}$	Data Hold Time for $\overline{\text{WRITE}}$	40		0		ns

**Notes:**

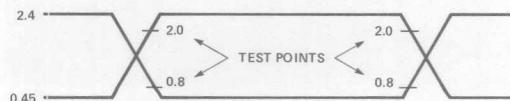
1. 8279,  $V_{CC} = +5\text{V} \pm 5\%$ ; 8279-5,  $V_{CC} = +5\text{V} \pm 10\%$ .
2. 8279,  $C_L = 100\text{pF}$ ; 8279-5,  $C_L = 150\text{pF}$ .

**Other Timings:**

Symbol	Parameter	8279		8279-5		Unit
		Min.	Max.	Min.	Max.	
$t_{\phi W}$	Clock Pulse Width	230		120		nsec
$t_{CY}$	Clock Period	500		320		nsec

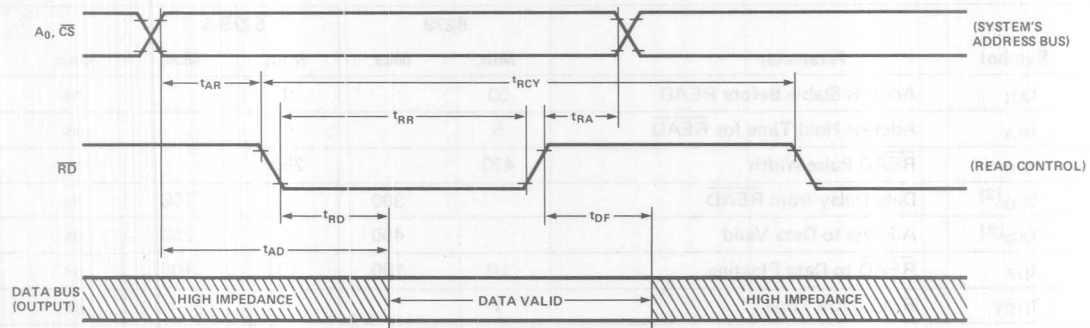
Keyboard Scan Time: 5.1 msec  
 Keyboard Debounce Time: 10.3 msec  
 Key Scan Time: 80  $\mu\text{sec}$   
 Display Scan Time: 10.3 msec

Digit-on Time: 480  $\mu\text{sec}$   
 Blanking Time: 160  $\mu\text{sec}$   
 Internal Clock Cycle: 10  $\mu\text{sec}$

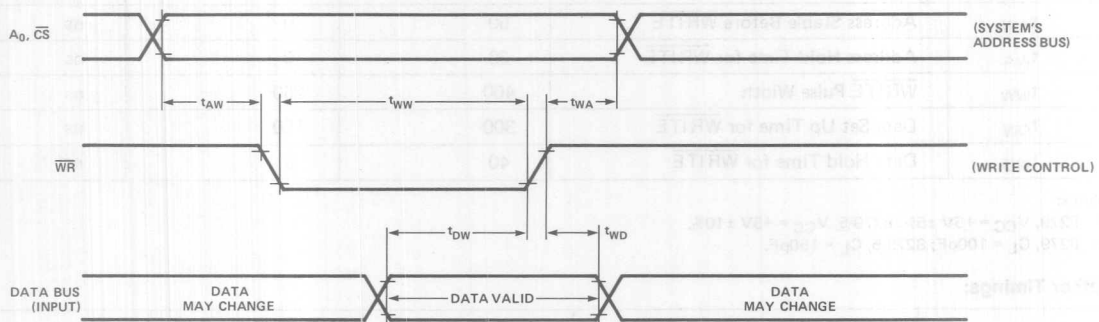
**Input Waveforms For A.C. Tests**

## WAVEFORMS

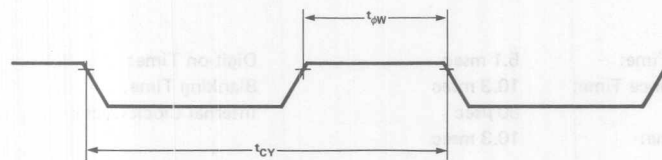
## Read Operation



## Write Operation



## Clock Input



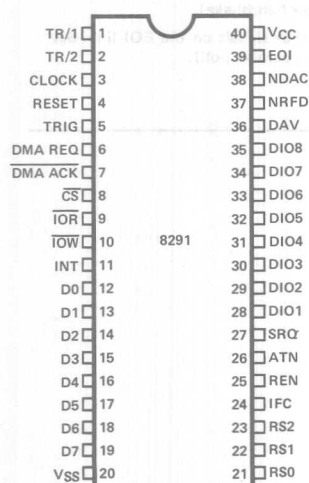


## 8291 GPIO TALKER/LISTENER

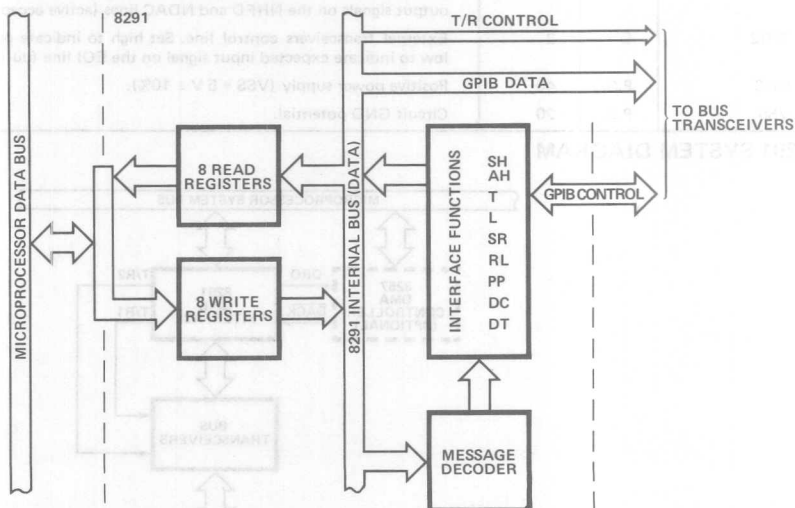
- Complete source and acceptor handshake.
- Complete talker and listener functions with extended addressing.
- Service request, parallel poll, device clear, device trigger, remote/local functions.
- Selectable Interrupts
- On chip primary and secondary address recognition.
- Automatic handling of addressing and handshake protocol.
- Provision for software implementation of additional features.
- Designed to interface 8-Bit Microprocessors (e.g., 8080, 8085, 8048) to an IEEE Standard 488 Digital Interface Bus.
- 16 Registers (8 Read, 8 Write), 2 for Data Transfer, the Rest for Interface Function Control, Status, etc.
- Directly Interfaces to External Transceivers for Connection to the GPIB Bus.
- Provides Three Addressing Modes, Allowing the Chip to be Addressed Either as a Major or a Minor Talker/-Listener with Primary or Secondary Addressing.
- DMA Handshake Provision Allows for Bus Transfers without CPU Intervention.
- Trigger Output Pin Allows for Triggering of any Device in the System Without CPU Intervention.
- On Chip EOS Message Recognition Facilitates Handling of Multi-Byte Transfers.

The 8291 GPIB TALKER/LISTENER is a microprocessor-controlled chip designed to interface 8-bit microprocessors (e.g., 8080, 8085, 8048) to an IEEE Standard 488 Instrumentation Interface Bus. It implements all of the Standard's talker/listener interface functions.

### PIN CONFIGURATION



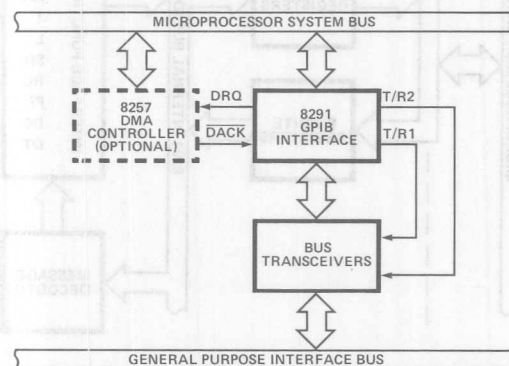
### BLOCK DIAGRAM



## PIN DESCRIPTION

Symbol	I/O	Pin No.	Function
D0-D7	I/O	12-19	Data bus port, to be connected to 8080, 8085, or 8048 data bus.
RS0-RS2	I	21-23	Register select inputs, to be connected to three non-multiplexed microprocessor address bus lines. Select which of the 8 internal read (write) registers will be read from (written into) with the execution of IOR (IOW).
$\overline{CS}$	I	8	Chip select. When low, enables reading from or writing into the register selected by RS0-RS2.
$\overline{IOR}$	I	9	Read strobe. When low, selected register contents are read by the CPU.
$\overline{IOW}$	I	10	Write strobe. When low, data is written into the selected register.
INT	O	11	Interrupt request to the microprocessor, set high for request and cleared when the appropriate register is accessed by the CPU.
DMA REQ	O	6	DMA request, normally low, set high to indicate byte output or byte input, in DMA mode; reset by DMA ACK.
DMA ACK	I	7	DMA acknowledge. When low, resets DMA REQ and selects data in/data out register for DMA data transfer (actual transfer done by IOR/IOW pulse).
TRIG	O	5	Trigger output, normally low; generates a triggering pulse corresponding to the GET command.
CLOCK	I	3	External clock input, used for internal time delays generator.
RESET	I	4	Reset input. When high, forces the device into an "Idle" (initialization) mode. The device will remain at "Idle" until released by the microprocessor.
DIO1-DIO8	I/O	28-35	8-bit GPIB data bus port, used for bidirectional data byte transfer between 8291 and GPIB bus via external line transceivers.
DAV	I/O	36	Data valid; GPIB bus handshake control line. Indicates the condition (availability and validity) of information on the DIO lines.
NRFD	I/O	37	Not ready for data; GPIB bus handshake control line. Indicates the condition of readiness of device(s) connected to the bus to accept data.
NDAC	I/O	38	Not data accepted; GPIB bus handshake control line. Indicates the condition of acceptance of data by the device(s) connected to the bus.
ATN	I	26	Attention; GPIB bus command line. Specifies how data on DIO lines are to be interpreted.
IFC	I	24	Interface clear; GPIB bus command line. Places the interface functions in a known quiescent state.
SRQ	O	27	Service request; GPIB bus command line. Indicates the need for attention and requests an interruption of the current sequence of events on the GPIB bus.
REN	I	25	Remote enable; GPIB bus command line. Selects (in conjunction with other messages) remote or local control of the device.
EOI	I/O	39	End or Identify; GPIB bus command line. Indicates the end of a multiple byte transfer sequence or, in conjunction with ATN, addresses the device during a polling sequence.
T/R1	O	1	External transceivers control line. Set high to indicate output data/signals on the DIO1-DIO8 and DAV lines and input signals on the NRFD and NDAC lines (active source handshake). Set low to indicate input data/signals on the DIO1-DIO8 and DAV lines and output signals on the NRFD and NDAC lines (active acceptor handshake).
T/R2	O	2	External transceivers control line. Set high to indicate output signals on the EOI line. Set low to indicate expected input signal on the EOI line (during parallel poll).
VCC	P.S.	40	Positive power supply ( $V_{SS} + 5V \pm 10\%$ ).
VSS	P.S.	20	Circuit GND potential.

## 8291 SYSTEM DIAGRAM



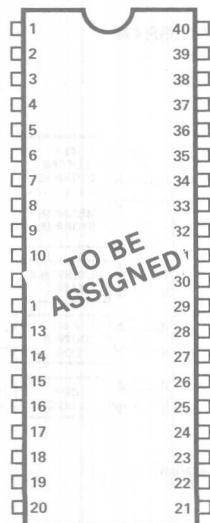
## 8292 GPIO CONTROLLER

### FEATURES:

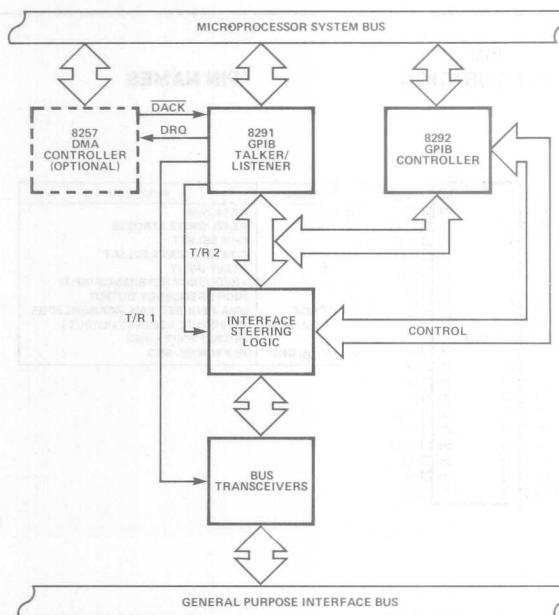
- Complete IEEE Standard 488 Controller Function.
- Interface Clear (IFC) Sending Capability Allows for Seizure of Control and/or Initialization of the Bus.
- Responds to Service Requests (SRQ).
- Sends (REN), Allowing Instruments to Switch to Remote Control.
- Complete Implementation of Transfer Control Protocol.
- Synchronous Control Seizure Prevents the Destruction of any Data Transmission in Progress.
- Connects with the 8291 to Form a Complete IEEE Standard 488 Interface Talker/Listener/Controller.

The 8292 GPIO CONTROLLER is a microprocessor-controlled chip designed to connect with the 8291 GPIO TALKER/LISTENER to implement the full IEEE Standard 488 controller function, including transfer control protocol.

### PIN CONFIGURATION



### 8291, 8292 SYSTEM DIAGRAM





## 8294 DATA ENCRYPTION UNIT

**PRELIMINARY**  
Notice: This is not a final specification. Some parametric limits are subject to change.

- Certified by National Bureau of Standards
- 80-Byte/Sec Data Conversion Rate
- 64-Bit Data Encryption Using 56-Bit Key
- DMA Interface
- 3 Interrupt Outputs to Aid in Loading and Unloading Data
- 7-Bit User Output Port
- Single 5V  $\pm$  10% Power Supply
- Peripheral to MCS-85<sup>TM</sup>, MCS-80<sup>TM</sup> and MCS-48<sup>TM</sup> Processors
- Implements Federal Information Processing Data Encryption Standard
- Encrypt and Decrypt Modes Available

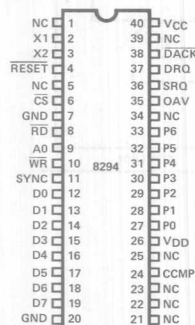
### DESCRIPTION

The Intel<sup>®</sup> 8294 Data Encryption Unit (DEU) is a microprocessor peripheral device designed to encrypt and decrypt 64-bit blocks of data using the algorithm specified in the Federal Information Processing Data Encryption Standard. The DEU operates on 64-bit text words using a 56-bit user-specified key to produce 64-bit cipher words. The operation is reversible: if the cipher word is operated upon, the original text word is produced. The algorithm itself is permanently contained in the 8294; however, the 56-bit key is user-defined and may be changed at any time.

The 56-bit key and 64-bit message data are transferred to and from the 8294 in 8-bit bytes by way of the system data bus. A DMA interface and three interrupt outputs are available to minimize software overhead associated with data transfer. Also, by using the DMA interface two or more DEUs may be operated in parallel to achieve effective system conversion rates which are virtually any multiple of 120 bytes/second. The 8294 also has a 7-bit TTL compatible output port for user-specified functions.

Because the 8294 implements the NBS encryption algorithm it can be used in a variety of Electronic Funds Transfer applications as well as other electronic banking and data handling applications where data must be encrypted.

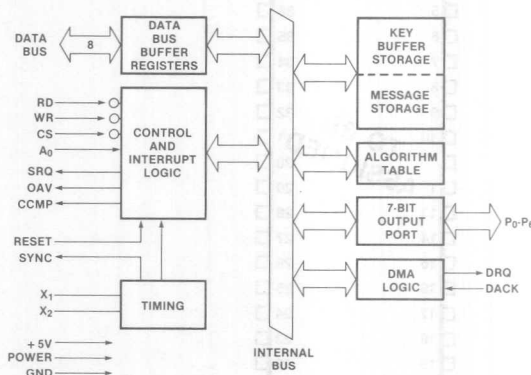
### PIN CONFIGURATION



### PIN NAMES

PIN NAME	FUNCTION
D7-D0	DATA BUS
RD, WR	READ, WRITE STROBES
CS	CHIP SELECT
A0	CONTROL/DATA SELECT
RESET	RESET INPUT
X1, X2	FREQUENCY REFERENCE INPUT
SYNC	HIGH FREQUENCY OUTPUT
DRQ, DACK	DMA REQUEST, DMA ACKNOWLEDGE
SRQ, OAV, CCMP	INTERRUPT REQUEST OUTPUTS
P6-P0	OUTPUT PORT LINES
VCC, VDD, GND	+5V POWER, GND

### BLOCK DIAGRAM



## COMMAND SUMMARY

### Enter New Key

OP CODE: 

0	1	0	0	0	0	0	0
---	---	---	---	---	---	---	---

  
MSB LSB

This command is followed by 8 data inputs which are retained in the key buffer (RAM) to be used in encrypting and decrypting data.

### Encode Data

OP CODE: 

0	0	1	1	0	0	0	0
---	---	---	---	---	---	---	---

  
MSB LSB

This command puts the 8294 into the encrypt mode.

### Decode Data

OP CODE: 

0	0	1	0	0	0	0	0
---	---	---	---	---	---	---	---

  
MSB LSB

This command puts the 8294 into the decrypt mode.

### Set Mode

OP CODE: 

0	0	0	0	A	B	C	D
---	---	---	---	---	---	---	---

  
MSB LSB

where:

- A is the OAV (Output Available) interrupt enable
- B is the SRQ (Service Request) interrupt enable
- C is the DMA (Direct Memory Access) transfer enable
- D is the CCMP (Conversion Complete) interrupt enable

This command determines which interrupt outputs will be enabled. A "1" in bits A, B, or D will enable the OAV, SRQ, or CCMP interrupts respectively. A "1" in bit C will allow DMA transfers. When bit C is set the OAV and SRQ interrupts should also be enabled (bits A,B = 1). Following the command in which bit C, the DMA bit, is set the 8294 will expect one data byte to specify the number of 8-byte blocks to be converted using DMA.

### Write to Output Port

OP CODE: 

1	P <sub>6</sub>	P <sub>5</sub>	P <sub>4</sub>	P <sub>3</sub>	P <sub>2</sub>	P <sub>1</sub>	P <sub>0</sub>
---	----------------	----------------	----------------	----------------	----------------	----------------	----------------

  
MSB LSB

This command causes the 7 least significant bits of the command byte to be latched as output data on the 8294 output port.

## FUNCTIONAL DESCRIPTION

In non-DMA mode, the conversion sequence is as follows:

1. A mode command is issued to enable the desired interrupt outputs.
2. A new key command is issued followed by 8 data inputs to initialize the key. Each byte must have odd parity.
3. The encrypt data or decrypt data command is issued to set the DEU in the desired mode.

After this, data conversions are made by writing 8 data bytes and then reading back 8 converted data bytes. Any of the above commands may be issued between data conversions to change the basic operation of the DEU; e.g., a decrypt data command could be issued to change the DEU from encrypt mode to decrypt mode without changing either the key or the interrupt outputs enabled.

## COMMAND AND DATA TRANSFER

Four internal registers are addressable by the master: 2 for input, 2 for output. Access and function of these registers are described below.

RD	WR	CS	A <sub>0</sub>	Register
1	0	0	0	Data input buffer
0	1	0	0	Data output buffer
0	1	0	1	Status output buffer
1	0	0	1	Command input buffer
X	X	1	X	Don't care

**Data Input Buffer** — Data written to this register is interpreted as part of a key, as data to be encrypted/decrypted, or as a DMA block count, depending on the command sequence preceding the write.

**Data Output Buffer** — Data read from this register will be the output of the encrypter/decrypter function.

**Status Output Buffer** — DEU status is available in this register at all times.

STATUS BIT:	7	6	5	4	3	2	1	0
FUNCTION:	XXX	XXX	XXX	KPE	HS	DEC	IBF	OBF

**OBF** — Output buffer full; OBF = 1 indicates that the output buffer contains encrypter/decrypter output data. It is set false when the data is read.

**IBF** — Input buffer full; IBF is set true when a command or data is written to the input buffer. The DEU sets this flag false when it has accepted the input byte. No data should be written when IBF = 1.

**DEC** — Decode; indicates whether the DEU is in encrypt or decrypt mode. Decrypt: DEC = TRUE; Encrypt: DEC = FALSE.

**HS** — Handshake flag; this flag is used in the data transfer protocol.

**KPE** — Key Parity Error; after a new key has been entered, the DEU will use this flag in conjunction with the HS flag to indicate correct or incorrect parity.

**Command Input Buffer** — Commands to the DEU are written to this register.

## INTERFACE TIMING

Figures 5 through 8 illustrate recommended protocol sequences and timing for transferring commands and data between the master processor and the 8294.

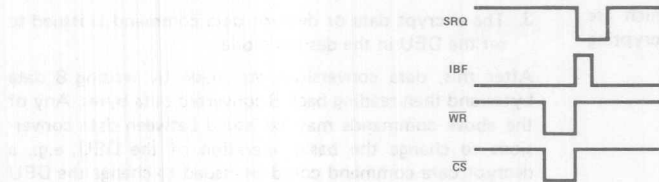


Figure 5. Single Byte Command

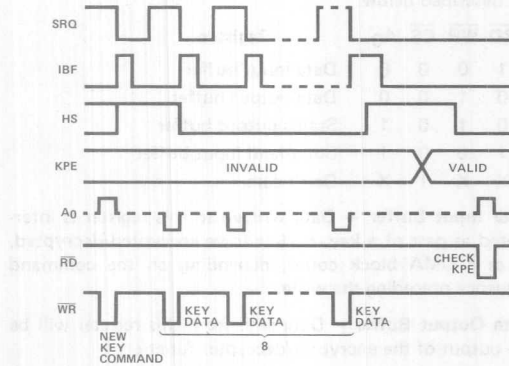


Figure 6. New Key Command

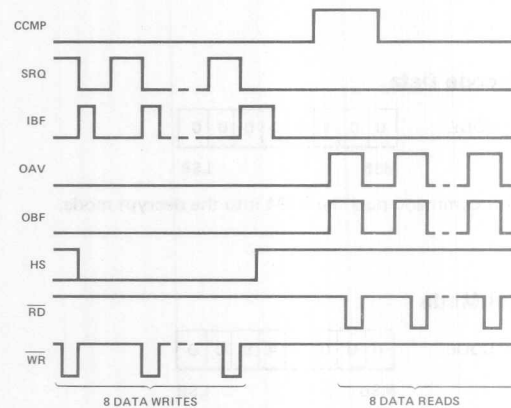


Figure 7. Encode/Decode data

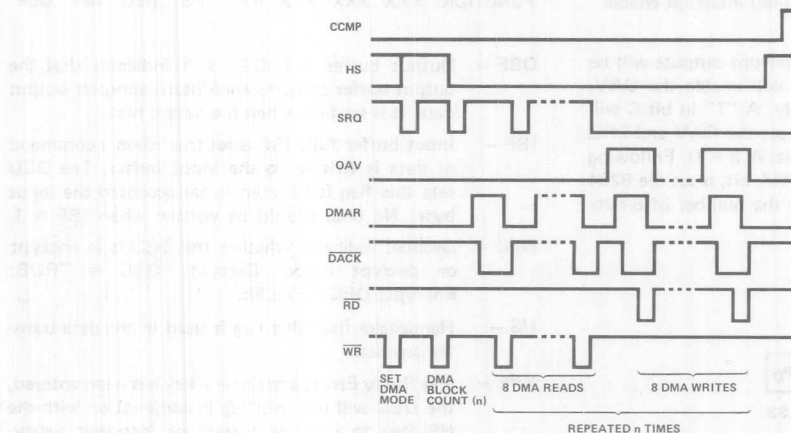


Figure 8. DMA Sequence

## MASTER/SLAVE INTERFACE

Figures 1 through 4 illustrate four interface configurations used in Master/Slave data transfers. In all cases SRQ will be true (if enabled) and IBF will be false when the DEU is ready to accept data or commands.

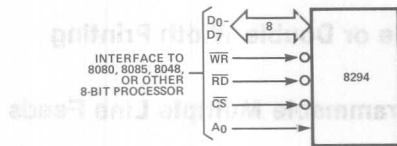


Figure 1. Polling Interface

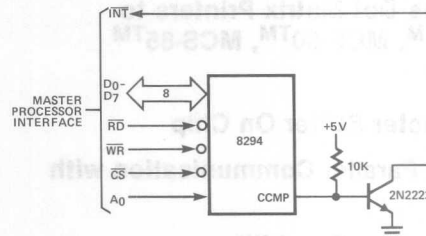


Figure 2. Single Interrupt Interface

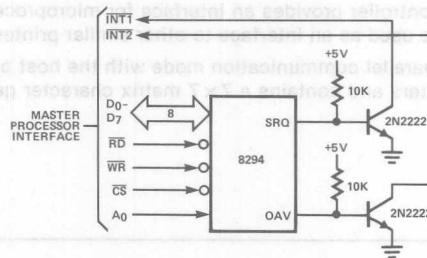
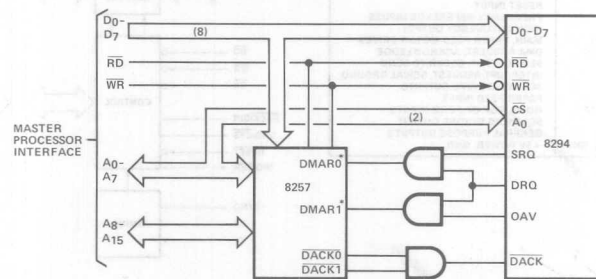


Figure 3. Dual Interrupt Interface



\*DMAR0 IS FOR MEMORY TO DEU DATA TRANSFER  
DMAR1 IS FOR DEU TO MEMORY DATA TRANSFER

Figure 4. DMA Interface

# DOT MATRIX PRINTER CONTROLLER

ADVANCE  
INFORMATION  
Characteristics are subject to change without notice.

- Interfaces Dot Matrix Printers to MCS-48™, MCS-80™, MCS-85™ Systems
- 40 Character Buffer On Chip
- Serial or Parallel Communication with Host
- DMA Transfer Capability
- Programmable Character Density (10 or 12 Characters/Inch)

- Programmable Print Intensity
- Single or Double Width Printing
- Programmable Multiple Line Feeds
- 3 Tabulations
- 2 General Purpose Outputs

The Intel® 8295 Dot Matrix Printer Controller provides an interface for microprocessors to the LRC 7040 Series dot matrix impact printers. It may also be used as an interface to other similar printers.

The chip may be used in a serial or parallel communication mode with the host processor. Furthermore, it provides internal buffering of up to 40 characters and contains a 7 x 7 matrix character generator accommodating 64 ASCII characters.

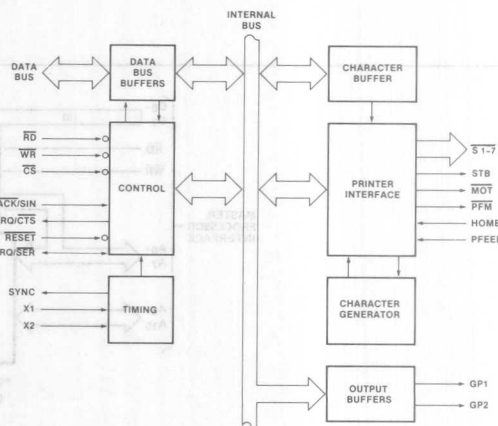
## PIN CONFIGURATION

PFEED	1	40	VCC
X1	2	39	HOME
X2	3	38	DACK/ISIN
RESET	4	37	DRO/CTS
NC	5	36	IRO/ISER
CS	6	35	MOT
GND	7	34	STB
RD	8	33	S7
VCC	9	32	S6
WR	10	31	S5
SYNC	11	30	S4
D0	12	29	S3
D1	13	28	S2
D2	14	27	S1
D3	15	26	VDD
D4	16	25	NC
D5	17	24	GP1
D6	18	23	GP2
D7	19	22	TOP
GND	20	21	PFM

## PIN NAMES

PIN NAME	FUNCTION
D0-D7	DATA BUS
RD, WR	READ, WRITE STROBES
CS	CHIP SELECT
RESET	RESET INPUT
X1, X2	FREQUENCY REFERENCE INPUTS
SYNC	HIGH FREQUENCY OUTPUT
MOT, PFM	MAIN, PAPER FEED MOTOR DRIVES
DRO, DACK	DMA REQUEST, ACKNOWLEDGE
SIN, CTS	SERIAL INPUT, CLEAR-TO-SEND
IRO/ISER	INTERRUPT REQUEST, SERIAL GROUND
S1-S7	SOLENOID DRIVE OUTPUTS
PFEED	PAPER FEED INPUT
HOME, TOP	HOME, TOP-OF-FORM INPUTS
STB	SOLENOID STROBE OUTPUT
GP1, GP2	GENERAL PURPOSE OUTPUTS
VCC, VDD, GND	+5V POWER, GND

## BLOCK DIAGRAM

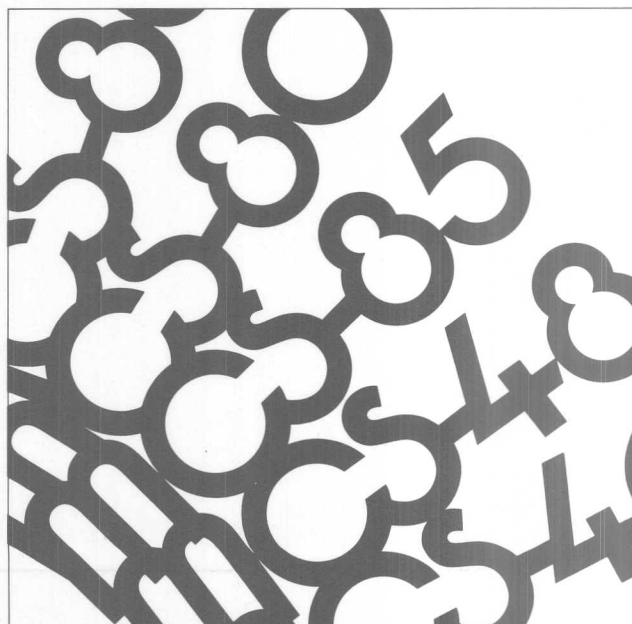


Printer Control with the UPI 11

by Linda Smith

## SECTION 2

# PERIPHERAL APPLICATION NOTES



# Printer Control with the UPI-41

by Lionel Smith

---

INTRODUCTION.....	2-3
THE LRC PRINTER.....	2-3
INTERFACE SIGNALS.....	2-4
TIMING .....	2-8
SOFTWARE .....	2-8
DETAILS OF THE BUFFER MANAGER.....	2-9
PRINTER SERVICE ROUTINES.....	2-11
CONCLUSION .....	2-14
APPENDIX .....	2-16



puter designed to be used as a universal peripheral interface device in a microcomputer system. The device is based on a completely self-contained 8-bit microcomputer with program memory, data memory, CPU, I/O, event timer, and clock oscillator, in a single 40-pin package. A bus interface is included which enables the UPI-41 to be used as a peripheral controller in MCS-48, MCS-80, MCS-85 and other 8-bit microcomputer families. The device is designed for keyboard scanning, printer control, display multiplexing and similar applications which involve interfacing peripheral devices to microcomputer systems.

The UPI-41 is fabricated with N-channel MOS technology and requires only a single 5-volt supply for operation. It has 1K words of program memory and 64 words of data memory on-chip. Both ROM (8041) and EPROM (8741) versions are available and the two are completely pin compatible. The instruction set of the UPI-41 is almost identical to that of the MCS-48. A single byte data register on the UPI-41 interfaces directly to an 8-bit master processor bus to handle asynchronous data transfer to and from the master system. A separate 4-bit register is used to indicate the status of data transfer. Two 8-bit TTL-compatible I/O ports plus two single-bit test inputs are available. I/O can be expanded further by using the 8243 I/O expander device. A separate register in the UPI-41 is used as an event counter or interval timer.

Because it is a complete microcomputer, the UPI-41 provides more power and flexibility than conventional LSI interface devices. For instance, the UPI-41 can be programmed as a peripheral interface for any of the low-cost drum or dot matrix printers currently on the market. In addition to controlling the printer, the UPI-41 can handle zero suppression, limit-checking, formatting and other computations, thereby unburdening the master processor. This type of distributed intelligence, made possible by the UPI-41, greatly enhances overall system capability while reducing cost and development time.

This application note describes how the UPI-41 can be used to implement an interface to a matrix printer. The printer chosen is fairly typical of a large class of printers which minimize total system cost by reducing the mechanical content at the expense of more sophisticated electronic requirements. The UPI-41, with its high degree of capabil-

UPI-41 read the "Intel UPI-41 User's Manual" before proceeding in this document.

## THE LRC PRINTER

The LRC Model 7040 printer is a matrix printer manufactured by LRC Inc. of Riverton, Wyoming. Capable of printing up to 40 columns of alphanumeric information, this printer is mechanically simple and should be ideal for a variety of applications such as point of sale terminals and data logging. While this note concentrates on the Model 7040 printer, the techniques discussed should be applicable to a variety of similar printers which are currently available.

The printer (Figure 1) consists of four major sub-assemblies, the frame, the print head, the main drive, and the paper handling components. The frame is an aluminum extrusion which provides a suitable base for mounting the various components of the printer. The print head consists of seven solenoids which each drive stiff wires to impact the paper through the inked ribbon. At the solenoid end of the print head these wires are arranged in a circular fashion. Where these wires impact the printer, however, the wires are arranged in a vertical column. To see how this arrangement can be used to print alphanumeric characters refer to Figure 2. The figure shows a  $5 \times 7$  matrix of "dots". The columns are labeled C1 through C5; the rows are labeled as Row 1 through Row 7. Each row corresponds to one of the solenoid-driven wires. The entire print head assembly is moved left to right across the paper so that at  $T_1$  it is over C1, at  $T_2$  it is over C2, and so on. If the correct solenoids are activated at each of these times ( $T_1$ – $T_5$ ) then a character can be formed. Figure 2 shows the character "A" formed. At  $T_1$  solenoids one through five were active, at  $T_2$  solenoids four and six were active, and so on until the complete character was formed. The complete character is formed by choosing the correct pattern of active solenoids for each of five instants in time.

The print head is moved across the paper by the main drive. The main drive consists of a 24-pole synchronous motor which drives a rotating plastic drum. The drum has a spiral groove molded into it. A pin attached to the print head rests in this groove so that as the drum rotates at a constant speed the print head is driven back and forth across the paper. Printing is accomplished by controlling

the activation of the solenoids as the print head is driven from left to right across the paper. When the end of the print area occurs the spiral groove reverses the direction of the head motion. As the left-hand edge of the paper is reached a cam attached to the drum activates the HOME micro-switch and the groove again reverses the motion of the head. When the print head is again over the print area and travelling in the left to right direction the microswitch is deactivated. The printer controller uses the trailing edge of the signal generated by the microswitch to initiate the printing of a new line of information.

Paper feed is accomplished by a second synchronous motor which can be activated to feed paper through the mechanism. A switch is provided which is activated while the actual line feed is occurring. The control logic can use the trailing

edge of the signal generated by this switch to turn off the line feed motor. A version of the printer with automatic line feed is available.

## INTERFACE SIGNALS

The interface signals to the printer consists of a pair of wires for each solenoid, a pair of wires for each motor (main drive and line feed), a pair of wires returning the state of the HOME micro-switch, and a pair of wires returning the state of the LINEFEED microswitch.

The solenoids must be driven from a  $40 \pm 4$  volt source. The peak current is approximately 3.6A, the average current is approximately 0.5A. A circuit providing the required drive is shown in Figure 3. The output stage, consisting of the 2N6045 Darlington transistor, the 1N4002 catching diode, and the 20-ohm damping resistor, is the

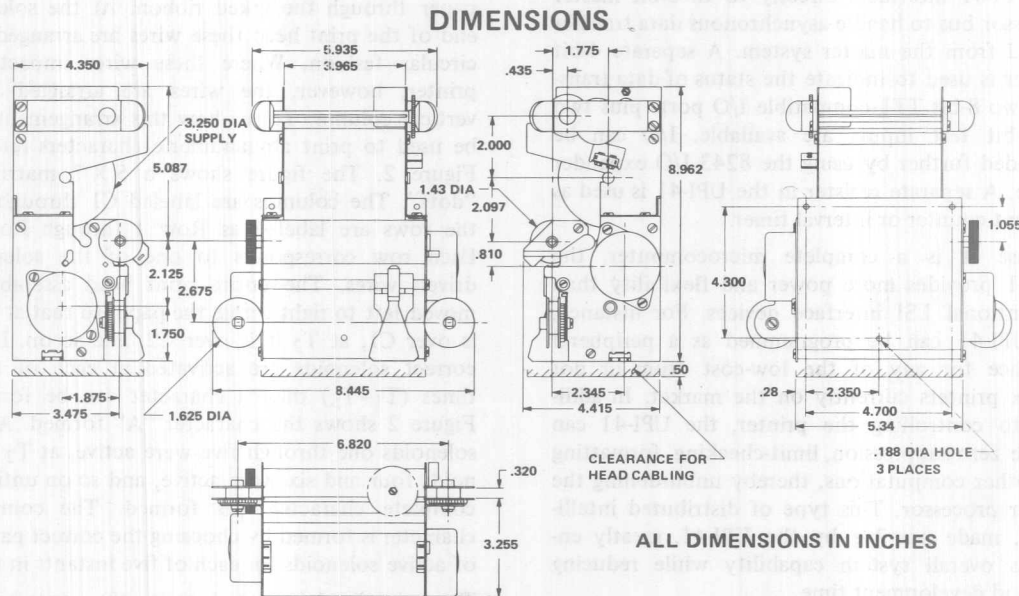


Figure 1. LRC Model 7040 Printer

one suggested by the manufacturer of the printer. The input stage is a discrete implementation of a DTL gate. Note that the base-emitter junction of the 2N6045 will protect the 2N2222A transistor from over-voltage on its collector. This circuit has several features which are important to the printer interface:

1. All solenoid power (including the power used to drive the base of the power transistor) is derived from the 40-volt supply.
2. Disconnecting the drivers from the UPI-41 or the loss of the 5-volt supply to the UPI-41 will result in the solenoids being turned off.

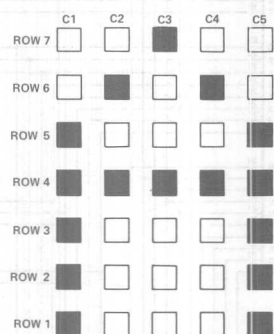


Figure 2. 5 x 7 Dot Matrix

The first feature of the drivers will minimize the impact of the printer and its interface on the 5-volt supply of the system. The second feature prevents the activation of the solenoids erroneously during power on/off cycles or during system checkout. This is an important point since the solenoids will be damaged if left activated continuously. (During the debug of the design described in this note fuses were added to the solenoid drivers to protect them from mishap.)

The two motors can each be driven as shown in Figure 4. The Monsanto MCS-6200 is an optically-coupled TRIAC which is ideal for driving the small synchronous motors in the printer. Coupled with a buffer this part provides a simple means of controlling the motor without sacrificing the isolation required for safe and reliable operation.

Figure 5 shows a UPI-41 used as an interface between an Intel® 8085 and an LRC Model 7040 printer. The drivers which have already been described have been used to interface the TTL outputs of the 8741 to the levels required by the printer. The two contact closure outputs from the printer (PAPERFEED and HOME) have been filtered and applied to the TEST0 and TEST1 inputs of the UPI-41. Bit 5 of output port 2 has been designated as an interrupt pin which will be used to request service from the 8085.

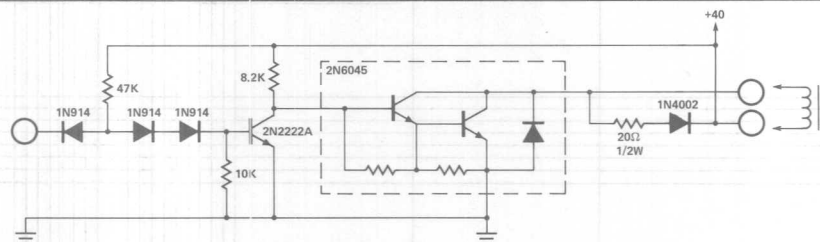


Figure 3. Solenoid Driver

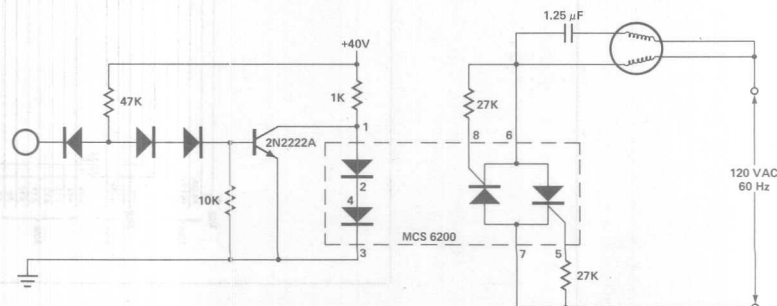


Figure 4. Motor Driver

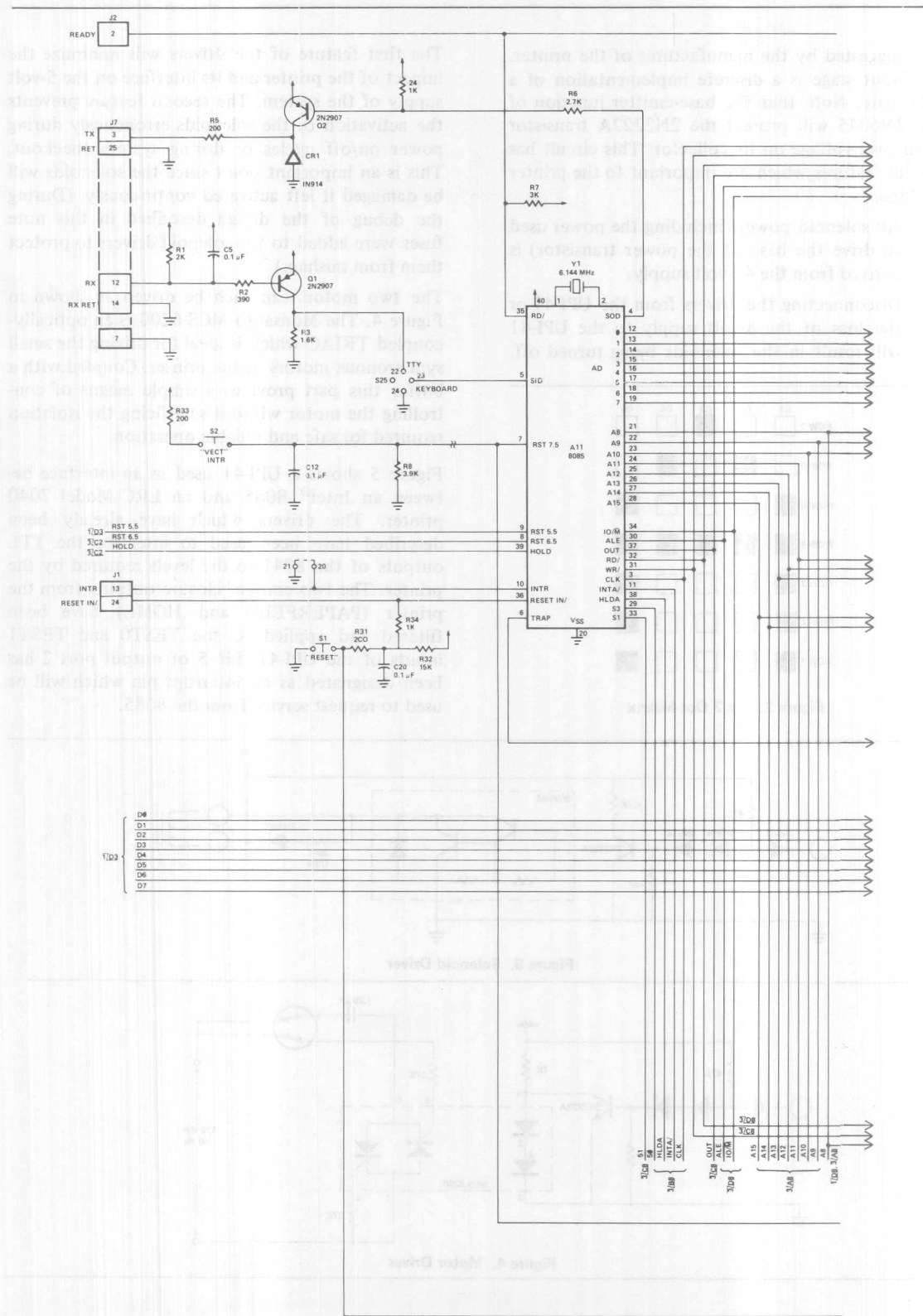


Figure 5. SDR-85 + UPI-41

The diagram illustrates the hardware configuration for the SDR-85 + UPI-41 system. It features three 8255 PPI chips (A14, A15, A16) and an 8205 decoder (A8). The system is organized into several sections, each corresponding to a specific port or function:

- PORT 0:** Connected to pins 24-31 of chip A14 and pins 24-31 of chip A15.
- PORT 1:** Connected to pins 32-39 of chip A14 and pins 32-39 of chip A15.
- PORT 8:** Connected to pins 24-31 of chip A16.
- PORT 9:** Connected to pins 32-39 of chip A16.
- PORT 21H:** Connected to pins 21-28 of chip A16.
- PORT 22H:** Connected to pins 29-36 of chip A16.
- PORT 23H:** Connected to pins 37-44 of chip A16.
- PORT 29H:** Connected to pins 21-28 of chip A17.
- PORT 2AH:** Connected to pins 29-36 of chip A17.
- PORT 2BH:** Connected to pins 37-44 of chip A17.

The diagram also shows the connection of a 3 MHz crystal oscillator (XTAL1, XTAL2) and a 3900 µF capacitor. At the bottom, there is a section for the motor driver, including a 220 kΩ resistor, a 1 µF capacitor, and a 110 VAC 60 Hz power source. The motor driver is connected to the T0, T1, and T2 pins of the 8255 chips.

## TIMING

The relative timing of the interface signals to the printer is shown in Figure 6. Actual printing commences when the main drive switch signal goes into the print ready state. This edge indicates that the print head is scanning across the paper in the left to right direction and that the printer is ready to start the actual printing of characters. When this edge occurs the UPI-41 must start transmitting pulses to each of the seven solenoids. The timing for these pulses is shown on the last line of Figure 6. A pulse of about 400 microseconds is used to generate a dot on the paper; a pause of about 900 microseconds between these pulses satisfies the duty cycle restrictions of the solenoids and provides a space between dots. Since the printer does not provide any feedback to the UPI-41 which would indicate the position of the print head, it is necessary for the UPI-41 to decide when to fire each solenoid based on timing information it maintains internally. The specifications of the printer allow 310 milliseconds for the print head to traverse the print area. The maximum repetition rate at which the solenoids can be fired is once every 1.3 milliseconds. The maximum number of dots that can be printed in the available print area is then  $310/1.3 = 238$ . After the last dot has been printed the line feed motor can be activated. The motor should remain activated until the line feed switch makes the off to on to off transition; this takes about 200 milliseconds. After the line feed motor is deactivated the next time of interest is when the main drive signal goes to the inactive state. At this point the printing of a complete line, including the necessary line feed, has been accomplished and the UPI-41 must prepare itself for the reactivation of the main drive switch. The activation of this switch will indicate that the printing of the next line can commence.

## SOFTWARE

The software system necessary to drive the LRC printer can be thought of as two main parts, each with an associated data structure. A block diagram of the system is shown in Figure 7. All the items shown above the dotted line are associated with the BUFFER MANAGER (BMGR) program part. All items shown below the dotted line are associated with a PRINTER SERVICE ROUTINE (PSR).

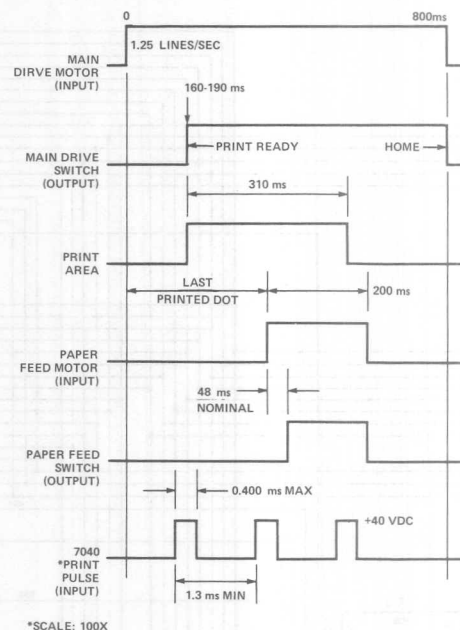


Figure 6. Printer Timing

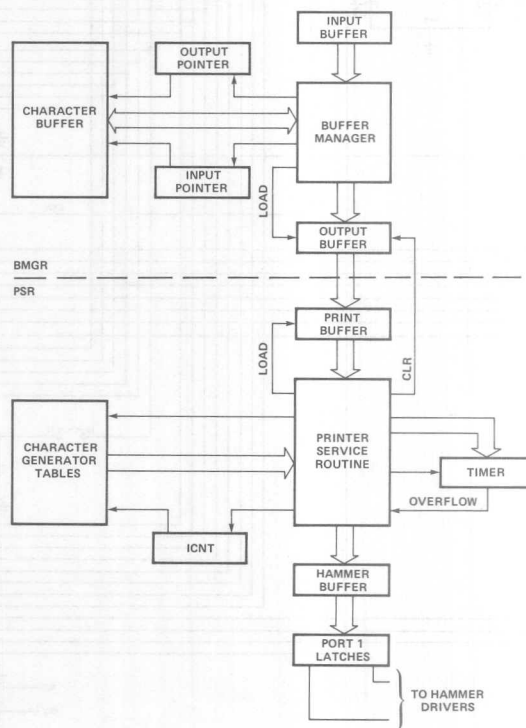


Figure 7. Software Block Diagram

The BUFFER MANAGER is responsible for all interaction with the master processor (i.e., the 8085 in Figure 5). The data structure associated with BMGR is a 40-character buffer which is used to store the characters as they are received from the master processor. BMGR maintains two pointers which are used to access the buffer; these pointers are shown as INPUT POINTER and OUTPUT POINTER in the diagram and are implemented as UPI-41 registers  $R_0$  and  $R_1$ , respectively. The input pointer (INPNT) is kept pointing to the last character loaded into the buffer, the output pointer (OUTPNT) is kept pointing to the next character to be printed. BMGR has two major interfaces, the INPUT BUFFER, which is used to communicate with the master processor, and the register shown in the figure as OUTPUT BUFFER. This register, which is implemented with register  $R_3$  of the UPI-41, is used to communicate with the printer service routine (PSR). A character to be printed is placed in the output buffer (OBUF). When PSR is ready to print the character it moves it from OBUF to its own buffer (PBUF) which is labeled as PRINT BUFFER in the diagram. After the character is moved the output buffer is overwritten by a predetermined value which indicates that PSR has accepted the character. BMGR will load a character into the output buffer only if it currently is equal to this value.

The printer service routine utilizes the TIMER to keep track of the current position of the print head. At the appropriate times it causes the solenoid drivers to be pulsed so that the character stream it sees in PBUF is printed. Based on the contents of PBUF and the contents of ICNT, which indicates the active column of the current character, PSR looks up the appropriate column data to be printed in the character generator tables. This data is stored in the HAMMER BUFFER until the precise time that it should be presented to the hammer drivers via the I/O bits in PORT 1. ICNT and the HAMMER BUFFER are implemented as UPI-41 registers 5 and 7, respectively.

#### DETAILS OF THE BUFFER MANAGER

Before BMGR can be discussed in detail, the manner in which it utilizes the character buffer must be understood. Figure 8 shows the operation of the buffer while two lines of data are input to the UPI-41 and subsequently printed. In order to keep the discussion manageable, this figure is drawn as if the printer were capable of printing only four

characters per line. The two lines of characters to be printed are:

ABCD  
1234

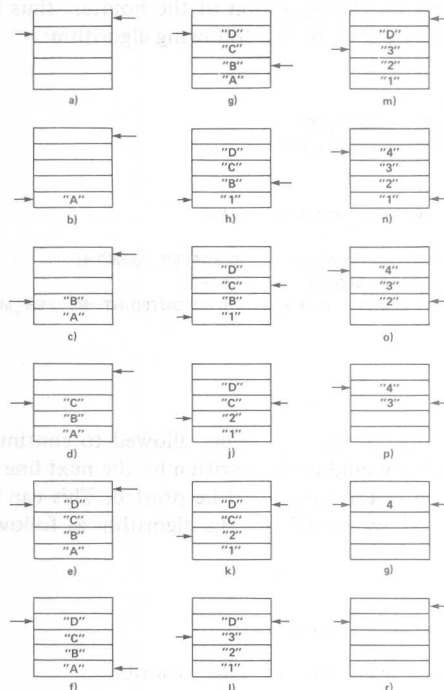


Figure 8. Buffer Operation

It should be noted that the buffer contains 5 bytes, one more than the number of print positions. The extra byte is a "phantom address" which, when pointed to by the output pointer, indicates that the section of BMGR which services the printer service routine is inactive. This state must be allowed because the actual print operation cannot begin until the complete line has been input to the buffer. If this rule were not enforced, some under-run protocol would have to be established to handle the situation of the input stream from the master processor failing to keep up with the print head.

Figure 8a shows the buffer in its initial state. The input pointer is set to the last real position in the buffer and the output pointer is set to the phantom position. Figures 8b through 8f show the operation of the pointers as the characters "A", "B", "C", and "D" are loaded. In each case the

input pointer is incremented to point to the next available location and then that location is loaded with the character. The position of the output pointer is not changed until the last position of the buffer has been loaded. When this occurs, the output pointer is set to point at the first character of the buffer. The operation of the pointers thus far can be described by the following algorithm:

```
INITIAL:
  INPOINT:=BUFFER_MAX;
  OUTPOINT:=BUFFER_MAX+1;
  ...
  LOOP:
    IF CHARACTER_AVAILABLE THEN
      BEGIN
        INPOINT:=(INPOINT+1) MOD BUFFER_LENGTH;
        BUFFER(INPOINT):=CHARACTER;
        IF INPOINT=BUFFER_MAX THEN OUTPOINT:=BUFFER_MIN;
      END;
      GOTO LOOP;
    END;
  END;
```

Obviously, if this loop were allowed to continue, the buffer would be overwritten by the next line of text before the first could be printed. This can be prevented by modifying the algorithm as follows:

```
...
  LOOP:
    IF CHARACTER_AVAILABLE THEN
      BEGIN
        TEMP:=(INPOINT+1) MOD BUFFER_LENGTH;
        IF TEMP<>OUTPOINT THEN
          BEGIN
            INPOINT:=TEMP;
            BUFFER(INPOINT):=CHARACTER;
            IF INPOINT=BUFFER_MAX THEN OUTPOINT:=BUFFER_MIN;
          END;
        END;
        GOTO LOOP;
      END;
    END;
```

This modification will “freeze the action” at Figure 8f until the output pointer is incremented. When this occurs the input procedure will immediately load the input data over the character that was just printed (assuming that data is available to the procedure at a higher rate than can be printed). The defined interface with the printer service routine allows a character to be removed from the buffer and placed in the output buffer whenever the output buffer contains the value placed there by the PSR, indicating that it has accepted the character that was previously in the output buffer. If this value is called EMPTY\_FLAG then the complete buffer handling procedure can be defined as follows:

```
INITIAL:
  INPOINT:=BUFFER_MAX;
  OUTPOINT:=BUFFER_MAX+1;
  ...
  LOOP:
    IF CHARACTER_AVAILABLE THEN
      BEGIN
        TEMP:=(INPOINT+1) MOD BUFFER_LENGTH;
        IF TEMP<>OUTPOINT THEN
          BEGIN
            INPOINT:=TEMP;
            BUFFER(INPOINT):=CHARACTER;
            IF INPOINT=BUFFER_MAX THEN
              OUTPOINT:=BUFFER_MIN;
            END;
            IF OUTPUT_BUFFER=EMPTY_FLAG THEN
              BEGIN
                IF OUTPOINT<=BUFFER_MAX THEN
                  BEGIN
                    OUTPUT_BUFFER:=BUFFER(OUTPOINT);
                    OUTPOINT:=OUTPOINT+1;
                  END;
                END;
              END;
            END;
            GOTO LOOP;
          END;
        END;
```

Examination of Figures 8g through 8r will show how this algorithm maintains the buffer. If there is an open position and a character is available, it is placed in the buffer. When a complete line is in the buffer, printing is initialized by setting the output pointer to BUFFER\_MIN. As the last character of a line is printed, the output pointer is incremented to point at the “phantom location” until the next line is completely entered. It should also be noted that if the input stream is faster than the print operation, then after the last character of a line is printed only one character need be input before printing can resume (see Figures 8l, m, and n). Frame r shows that after all available characters have been printed the state of the buffer is the same as it is initially. This is obviously a desirable feature.

The flowcharts for the complete BUFFER MANAGER are shown in Figures 9a and 9b. The corresponding code can be found starting at label BMGR of the program listings (see appendix). The flowcharts follow the algorithm that has been discussed very closely. Some additions have been made to implement logic not associated with the buffer. The first difference is that when a byte is in the input buffer it is tested to determine whether it is a command byte or a data character before further action is taken. Only two commands are recognized; one to set, and one to reset, the internal interrupt enable flag. This flag, which is

implemented as bit zero of PORT2 determines whether or not the UPI-41 will assert an interrupt to the master processor when it is able to accept a new character. Two additional deviations can be noted in Figure 9a; the first is that the motor of the printer will be turned on whenever a data character is received, the second is that if an end of line code (i.e., an ASCII line feed) is received, then, instead of storing it in the buffer, a mode is entered which fills the remaining buffer locations with space characters. This mode is enabled by bit one of PORT2. Note that utilizing otherwise unused bits of PORT2 for program status allows convenient testing and setting by the software and also enables external monitoring of the program operation.

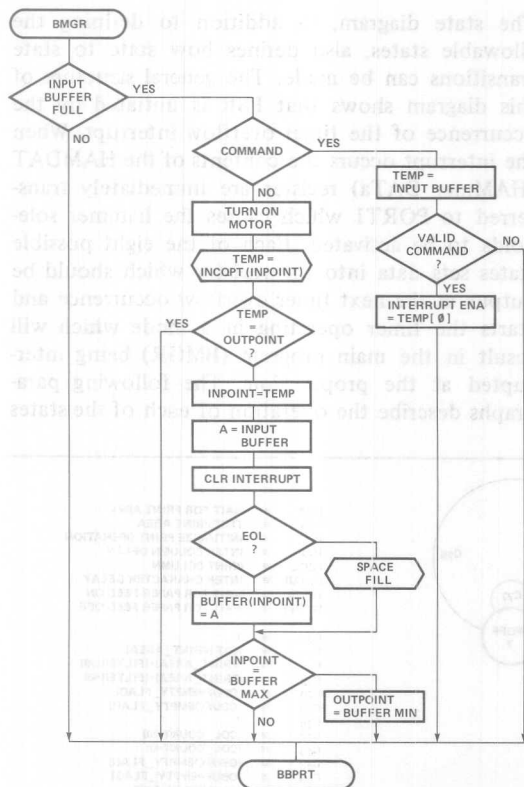


Figure 9a. Buffer Manager Flowchart

The last addition to the algorithm can be seen in Figure 9b where instead of going directly back to the start of the program after servicing the printer, a test is made to determine if the interrupt to the master processor should be asserted. This interrupt is set if the enable bit is set and there is also room in the buffer for at least one more character. After this test, control is passed back to the beginning of BMGR.

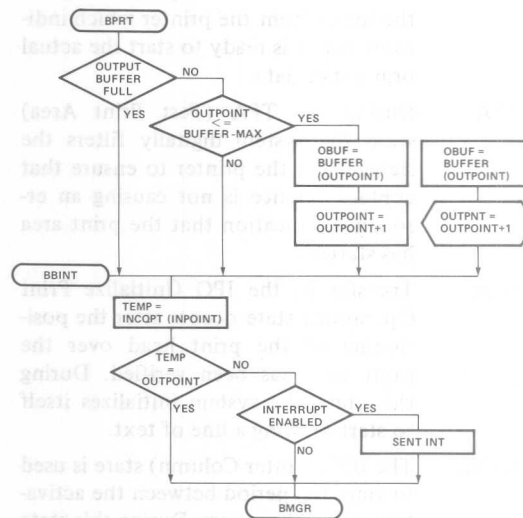


Figure 9b. Buffer Manager Flowchart

## PRINTER SERVICE ROUTINES

The Printer Service Routine must convert the characters given to it by the Buffer Manager into an appropriately timed stream of pulses to the solenoids. Because the PSR is extremely time-dependent, it was implemented as an interrupt-driven routine which is given control when the timer overflow occurs. This allows exact timing of the solenoid firings without requiring software delay loops. If the timing had been generated by such loops, synchronization would have been lost when the delay loops were interrupted in order to service the master processor.

If a hardware design of a controller for the printer were being undertaken, a convenient place to start would be to generate a state transition diagram which shows all the states that can be entered and how control can transfer from state to state. This hardware design technique is often useful in software design and was, in fact, used to develop the PSR. The state diagram of the PSR is shown in Figure 10. A total of eight states are necessary to implement the printer control function. Before discussing this diagram further, each of these states must be defined.

**WPA:** The WPA (Wait for Print Area) state is the state in which the system waits for the input from the printer which indicates that it is ready to start the actual printing of data.

**TPA:** During the TPA (Test Print Area) state the system digitally filters the signal from the printer to ensure that contact bounce is not causing an erroneous indication that the print area has started.

**IPO:** Transfer to the IPO (Initialize Print Operation) state occurs after the positioning of the print head over the print area has been verified. During this state the system initializes itself to start printing a line of text.

**ICOL:** The ICOL (Inter Column) state is used to time the period between the activation of the hammers. During this state the space between the dots of the characters is generated.

**PCOL:** During the PCOL (Print Column) state the hammers are energized if the particular character being printed requires a dot in the corresponding position.

**ICHAR:** The ICHAR (Inter Character) state is active between characters on a given line.

**WFON:** During the WFON (Wait for Feed On) state the system waits for the assertion of the feed pulse from the printer. This signal indicates that the process of feeding paper is occurring.

**WFOFF:** The system remains in the WFOFF (Wait for Feed Off) until the feed pulse goes inactive. This indicates that the required paper feed operation has been completed.

The state diagram, in addition to defining the allowable states, also defines how state to state transitions can be made. The general structure of this diagram shows that PSR is initiated by the occurrence of the timer overflow interrupt. When the interrupt occurs the contents of the HAMDAT (HAMmer DATA) register are immediately transferred to PORT1 which causes the hammer solenoids to be activated. Each of the eight possible states sets data into the register which should be output at the next timer overflow occurrence and starts the timer operating in a mode which will result in the main program (BMGR) being interrupted at the proper time. The following paragraphs describe the operation of each of the states

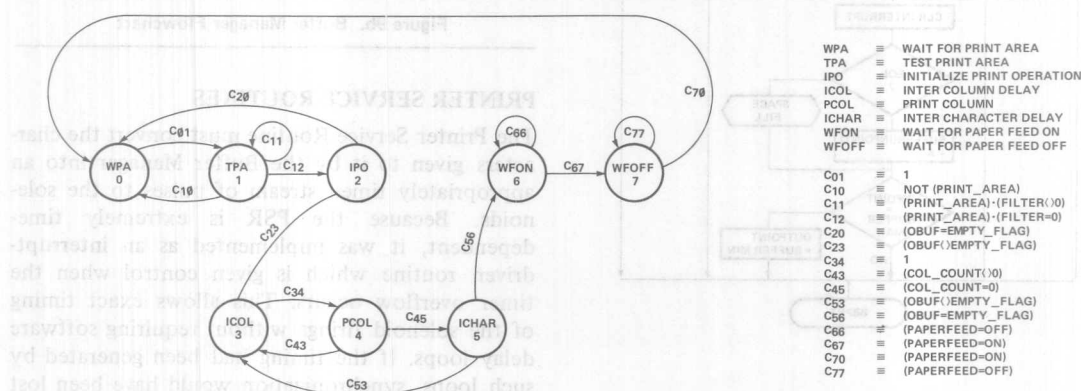


Figure 10. Print Control State Transition Diagram

in detail. The flowcharts of the routines can be found in Figure 11.

The WPA, CPA, and IPO states are all associated with the detection of the valid start of the print area. The WPA state sets the timer in the event count mode so that the edge of the print area signal can be detected, the CPA state digitally filters this input once it has been detected to ensure that noise has not caused a false input, and finally, the IPO state initializes the system to start the actual printing of data. The flowchart shows that the WPA state accomplishes the following actions:

1. Turns off the paper feed motor
2. Sets the filter count (for the CPA state)
3. Sets HAMDAT to zero
4. Sets STATE to one.

The timer is set to event count with an initial value of OFFH. This will cause a timer overflow interrupt the next time a negative transition occurs on the TEST1 input. Since this input is tied to the signal from the PRINT AREA switch, this interrupt should occur when the start of the print area is reached. The WPA state sets the STATE register to cause the TPA state to be entered when this interrupt occurs. Each time the TPA (Test Print Area) state is activated the software checks to ensure that the print area switch is in the proper state; if it is not, then all the actions of state zero are repeated (except turning off the motor), since a false start of print area has occurred. If the test reveals that the print area switch is in the proper state, then the filter count is reduced by one and the timer is started with an initial value of OFFH, the minimum attainable timer increment. The STATE register is set to repeat the TPA state unless the filter count has reached zero; when this occurs the IPO state is selected. The IPO state, which is responsible for the initialization of the actual print operation, first tests the output buffer register to determine if there is any data for it to print. If this test is unsuccessful the printer main drive motor is turned off, the TPA state is reinvoked and the timer is started in the event count mode so that it can detect the next start of print area. At first glance this seems somewhat fruitless since the event required cannot happen if the motor is not turning. By referring back to Figure 9, however, it can be seen that BMGR turns on the motor whenever it has a data character from the master computer. The reception of a character will always allow the PSR to find the next print area. If, when the IPO state makes its

test, there is data in the output buffer then the data is moved to the print buffer and the output buffer is set to the empty value. After this is accomplished, a counter is set to the number of columns to be printed per character (seven in this case — see comment by CGEN label in program listing), the STATE register is set to the ICOL state and the timer is set to time the intercolumn time. (The intercolumn time is the time that elapses between each possible column of the character.) Before exiting from this state the first column of data for the hammers is generated by the COLUMN routine and placed in the HAMDAT register.

The three states already discussed set the printer up so that it is ready to print. The next three states are repeated sequentially until the entire line of data has been printed. The ICOL state is probably the simplest of the states. When it is invoked the hammers have just been fired by the entry into the PSR. All that the ICOL state does is to set the timer to time the proper duration of the hammer strikes, clear the HAMDAT register, and set the STATE register to the PCOL state. The PCOL state, only slightly more complicated than the ICOL state, first decrements the column count. If the end of a character is detected (count equal zero), the HAMDAT register is cleared and the STATE register is set to invoke the ICHAR state. If the end of a character is not detected then the COLUMN routine is again used to determine the next data to be sent to the hammers and the ICOL state is reinvoked. When the ICOL state is active two things can happen, depending on whether there is more data to print. If there is data in the output buffer then a series of actions similar to those of the IPO state occur to reinitialize the printing of a character; if there is no more data in the line then the paper feed motor is turned on, HAMDAT is cleared, and the STATE register is set to the WFON state. The timer is set for approximately one millisecond so that the state of the paper feed switch can be sampled periodically by the WFON and WFOFF states.

The WFON and WFOFF states continue to set the timer to the one millisecond sample rate, the WFON state reinvokes itself until the paper feed switch input is detected and then it invokes the WFOFF state. The WFOFF state reinvokes itself until the paper feed switch is detected in the off state and then invokes the WPA state. The sole purpose of the WFON and WFOFF states is to ensure that an off to on to off transition occurs on



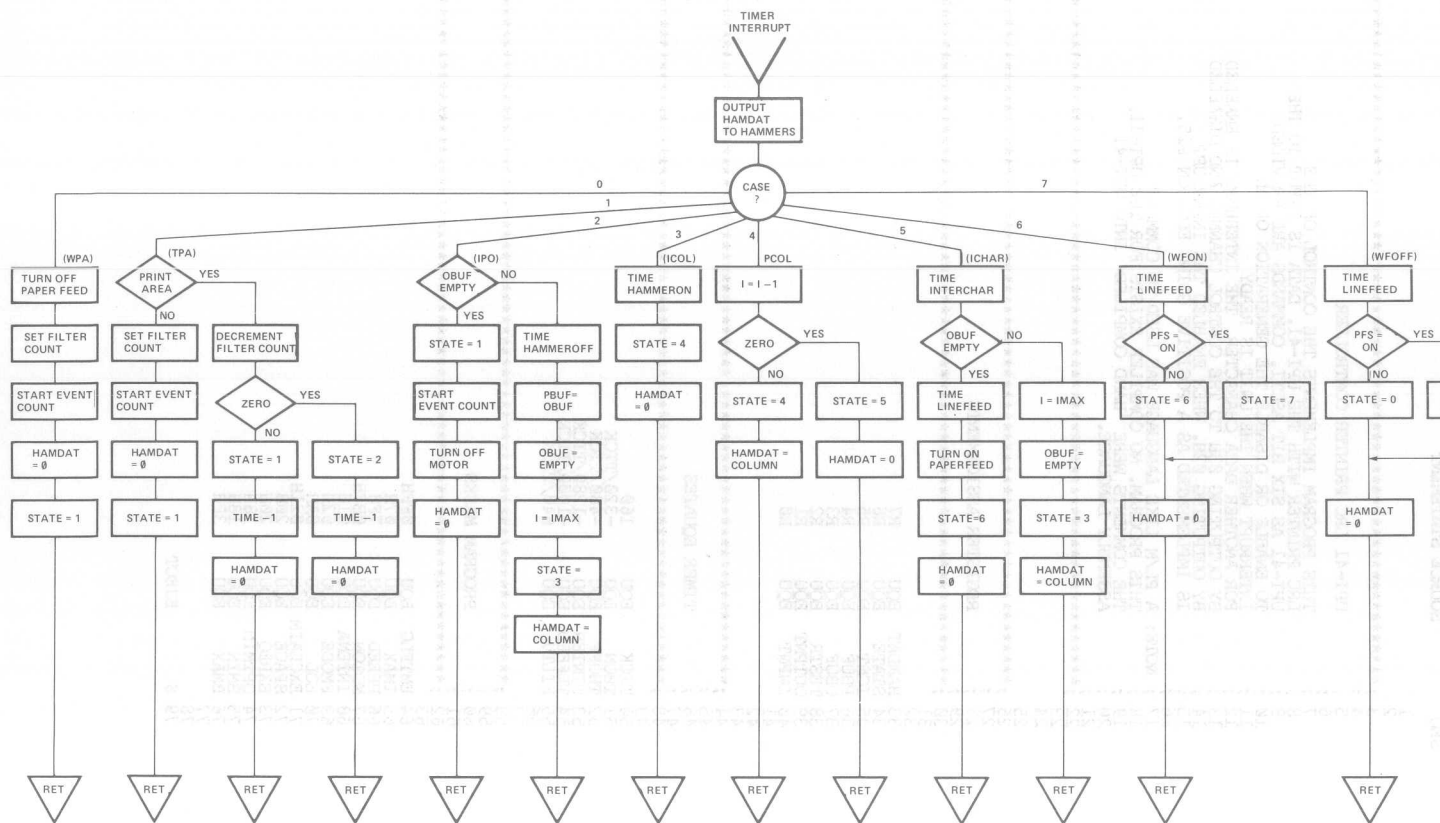


Figure 11. PSR Flowchart

# APPENDIX

ISIS-II 8048 ASSEMBLER, V1.1

LRC PRINTER CONTROLLER 7/14/7

LOC	OBJ	SEQ	SOURCE STATEMENT
		1	
		2	
		3	*****
		4	
		5	
		6	UPI-41 LRC PRINTER CONTROLLER
		7	
		8	THIS PROGRAM IMPLEMENTS THE CONTROL OF THE
		9	LRC PRINTER WITH THE UPI-41. DATA IS INPUT TO THE
		10	UPI-41 AS SIX BIT ASCII. COMMANDS ARE PROVIDED
		11	TO ENABLE OR DISABLE THE GENERATION OF AN
		12	INTERRUPT WHEN THE UNIT IS READY
		13	FOR ANOTHER DATA CHARACTER. THE INTERRUPT IS ENABLED
		14	BY OUTPUTTING 03H TO THE CONTROL CHANNEL AND DISABLED
		15	BY OUTPUTTING 02H. WHEN ENABLED THE INTERRUPT
		16	IS IMPLEMENTED AS A POSITIVE GOING EDGE ON P25.
		17	
		18	NOTE: A PL/M LIKE LANGUAGE WAS USED TO COMMENT
		19	THIS PROGRAM. NO COMPILER EXISTS FOR THE UPI-41.
		20	THE COMMENTS WERE 'HAND COMPILED' INTO UPI-41
		21	ASSEMBLY LANGUAGE.
		22	*****
		23	
		24	
		25	
		26	*****
		27	
		28	REGISTER ASSIGNMENTS
		29	
		30	*****
		31	
		32	
0007		33	HAMDAT EQU R7
0006		34	STATE EQU R6
0005		35	ICNT EQU R5
0004		36	PBUF EQU R4
0003		37	OBUF EQU R3
0002		38	TESTR EQU R2
0001		39	OUTPNT EQU R1
0000		40	INPNT EQU R0
		41	
		42	
		43	
		44	*****
		45	
		46	TIMER EQUATES
		47	
		48	*****
		49	
00A0		50	TICK EQU 160
FFFE		51	THON EQU -320/TICK
FFFD		52	THOFF EQU -480/TICK
FFF8		53	TINTIR EQU -1280/TICK
FFFA		54	TLFEED EQU -1000/TICK
0004		55	FILTIV EQU 640/TICK
		56	
		57	
		58	*****
		59	
		60	PROGRAM MASKS
		61	
		62	*****
		63	
00FF		64	EMTFLG EQU 0FFH
0007		65	IMAX EQU 07H
007F		66	PFEED EQU 7FH
00BF		67	MOTCN EQU 0BFH
0001		68	INTENA EQU 01H
0002		69	FMODE EQU 02H
000A		70	EOL EQU 0AH
0021		71	EXCLAIM EQU 021H
0020		72	SPACE EQU 20H
0020		73	EXREQ EQU 20H
0018		74	OPTMIN EQU 18H
0018		75	BMIN EQU 18H
003F		76	BMAX EQU 3FH
		77	
		78	
		79	\$ EJECT

LOC	OBJ	SEQ	SOURCE STATEMENT
		80	*****
		81	*****
		82	*****
		83	START OF PROGRAM
		84	*****
		85	*****
		86	*****
		87	*****
		88	; INITIALIZE;
		89	; INITIALIZE AND GO TO
		90	; BMGR
0000	1416	91	RESET: ORG 00H
0002	3479	92	CALL CASE0
0004	25	93	CALL INIT
0005	2400	94	EN TCNTI
		95	JMP BMGR ; CODE MUST END AT LOC 6
		96	*****
		97	*****
		98	*****
		99	START OF INTERRUPT DRIVEN STATE MACHINE
		100	*****
		101	*****
		102	*****
		103	; DO;
		104	HAMMERS=HAMMERS\$DAT;
		105	DO CASE STATE;
0007	2F	106	TISR: XCH A,HAMDAT
0008	37	107	CPL A
0009	39	108	OUTL P1,A
000A	FE	109	MOV A,STATE
000B	030E	110	ADD A,#CBASE
000D	B3	111	JMPP @A
000E	16	112	CBASE: DB CASE0
000F	24	113	DB CASE1
0010	40	114	DB CASE2
0011	61	115	DB CASE3
0012	6B	116	DB CASE4
0013	7D	117	DB CASE5
0014	9E	118	DB CASE6
0015	AE	119	DB CASE7
		120	
		121	
		122	
		123	
		124	
		125	DO; /*CASE 0, FEEDING LINE */
		126	PAPERSFEED=OFF;
		127	STATE=1;
		128	ICNT=FILTIV
		129	WAIT(PRINT\$AREA);
		130	HAMMERS\$DATA=0;
		131	END; /* END OF CASE 0 */
0016	8A80	132	CASE0: ORL P2,#(NOT PFEEED)
0018	BE01	133	MOV STATE,#1
001A	BD04	134	MOV ICNT,#FILTIV
001C	23FF	135	MOV A,#-1
001E	62	136	MOV T,A
001F	45	137	STRT CNT
0020	2300	138	MOV A,#0
0022	2F	139	XCH A,HAMDAT
0023	93	140	RETR
		141	
		142	DO; /* CASE1, TESTING FOR PRINT AREA */
		143	IF T0=1 THEN
		144	DO;
		145	STATE=1;
		146	ICNT=FILTIV;
		147	WAIT(PRINT\$AREA);
		148	HAMMERS\$DATA=0;
		149	END;
0024	4632	150	CASE1: JNT1 C1ELS
0026	BE01	151	MOV STATE,#1
0028	BD04	152	MOV ICNT,#FILTIV
002A	23FF	153	MOV A,#-1
002C	62	154	MOV T,A
002D	55	155	STRT T
002E	2300	156	MOV A,#0
0030	043E	157	JMP C1END

LOC	OBJ	SEQ	SOURCE STATEMENT
		158	; ELSE DO;
		159	ICNT=ICNT-1;
		160	IF ICNT=0 THEN STATE=2 ELSE STATE=1;
		161	TIME (-1);
		162	HAMMERSDATA=0;
		163	END;
0032	BE02	164	C1ELS: MOV STATE,#2
0034	ED38	165	DJNZ ICNT,C1A
0036	BE01	166	MOV STATE,#1
0038	23FF	167	C1A: MOV A,#-1
003A	62	168	MOV T,A
003B	55	169	STRT T
003C	2300	170	MOV A,#0
		171	END; /*END OF CASE1 */
003E	2F	172	C1END: XCH A,HAMDAT
003F	93	173	RETR
		174	
		175	
		176	DO; /*CASE 2, INITIALIZE PRINT OPERATION */
		177	IF OBUF<>EMPTY\$FLAG THEN
		178	DO;
		179	TIME (HAMMERSOFF);
		180	PBUF=OBUF;
		181	OBUF=EMPTY\$FLAG;
		182	I=IMAX;
		183	STATE=3;
		184	HAMMERSDATA=COLUMN (PBUF,I);
		185	END;
0040	FB	186	CASE2: MOV A,OBUF
0041	D3FF	187	XRL A,#EMITFLG
0043	C655	188	JZ C2ELS
0045	23FD	189	MOV A,#THOFF
0047	62	190	MOV T,A
0048	55	191	STRT T
0049	FB	192	MOV A,OBUF
004A	AC	193	PBUF,A
004B	BBFF	194	OBUF,#EMITFLG
004D	BD07	195	MOV ICNT,#IMAX
004F	BE03	196	MOV STATE,#3
0051	54E0	197	CALL COLUMN
0053	045F	198	JMP C2END
		199	
		200	ELSE DO;
		201	STATE=1;
		202	WAIT (PRINT\$AREA);
		203	MOTOR=OFF;
		204	HAMMERSDATA=0;
		205	END;
0055	BE01	205	C2ELS: MOV STATE,#1
0057	23FF	206	MOV A,#-1
0059	62	207	MOV T,A
005A	45	208	STRT CNT
005B	8A40	209	ORL P2,#NOT MOTON
005D	2300	210	MOV A,#0
		211	END; /*END OF CASE 2 */
005F	2F	212	C2END: XCH A,HAMDAT
0060	93	213	RETR
		214	
		215	DO; /*CASE 3, HAMMER OFF CYCLE */
		216	TIME (HAMMERSON);
		217	HAMMERSDATA=0;
		218	STATE=4;
		219	END; /*END OF CASE 3 */
0061	23FE	220	CASE3: MOV A,#THON
0063	62	221	MOV T,A
0064	55	222	STRT T
0065	2300	223	MOV A,#0
0067	BE04	224	MOV STATE,#4
0069	2F	225	XCH A,HAMDAT
006A	93	226	RETR
		227	\$ EJECT

LOC	OBJ	SEQ	SOURCE STATEMENT	
		228		DO; /*CASE 4, PRINTING COL I OF CHAR */
		229		TIME (HAMMER\$OFF);
006B	23FD	230	CASE4: MOV A, #THOFF	
006D	62	231	MOV T, A	
006E	55	232	STRT T	
		233		I=I-1;
		234		IF I=0 THEN
		235		DO;
		236		STATE=5;
		237		HAMMER\$DATA=0;
		238		END
006F	ED77	239	DJNZ ICNT, C4ELS	
0071	BE05	240	MOV STATE, #5	
0073	2300	241	MOV A, #0	
0075	047B	242	JMP C4END	
		243		ELSE DO;
		244		STATE=3;
		245		HAMMER\$DATA=COLUMN (PBUF, I);
		246		END;
0077	BE03	247	C4ELS: MOV STATE, #3	
0079	54E0	248	CALL COLUMN	
		249		END; /* END OF CASE 4 */
007B	2F	250	C4END: XCH A, HAMDAT	
007C	93	251	RETR	
		252		DO; /*CASE 5, INTERCHARACTER SPACE */
		253		TIME (INTER\$CHAR);
007D	23F8	254	CASE5: MOV A, #TINTER	
007F	62	255	MOV T, A	
0080	55	256	STRT T	
		257		IF OBUF<>EMPTY\$FLAG THEN
		258		DO;
		259		PBUF=OBUF;
		260		OBUF=EMPTY\$FLAG;
		261		I=IMAX;
		262		STATE=3;
		263		HAMMER\$DATA=COLUMN (PBUF, I);
		264		END;
0081	FB	265	MOV A, OBUF	
0082	D3FF	266	XRL A, #EMTFLG	
0084	C692	267	JZ C5ELS	
0086	FB	268	MOV A, OBUF	
0087	AC	269	MOV PBUF, A	
0088	BBFF	270	MOV OBUF, #EMTFLG	
008A	BD07	271	MOV ICNT, #IMAX	
008C	BE03	272	MOV STATE, #3	
008E	54E0	273	CALL COLUMN	
0090	049C	274	JMP C5END	
		275		ELSE DO;
		276		TIME (LINESFEED);
		277		PAPER\$FEED=ON;
		278		STATE=6;
		279		HAMMER\$DATA=0;
		280		END;
0092	23FA	281	C5ELS: MOV A, #TLFEED	
0094	62	282	MOV T, A	
0095	55	283	STRT T	
0096	9A7F	284	ANL P2, #PFEED	
0098	BE06	285	MOV STATE, #6	
009A	2300	286	MOV A, #0	
		287		END; /* END OF CASE 5 */
009C	2F	288	C5END: XCH A, HAMDAT	
009D	93	289	RETR	
		290		
		291	\$ EJECT	

LOC	OBJ	SEQ	SOURCE STATEMENT
		292	DO; /*CASE 6, WAITING FOR FEED ON */
		293	TIME(LINESFEED);
		294	IF PFS=1 THEN
		295	DO;
		296	STATE=7;
		297	END;
009E	23FA	298	CASE6: MOV A, #TLFEED
00A0	62	299	MOV T, A
00A1	55	300	STRT T
00A2	26A8	301	C6ELS
00A4	BE07	302	MOV STATE, #7
00A6	04AA	303	JMP C6END
		304	;
		305	ELSE DO;
		306	STATE=6;
		307	END;
00A8	BE06	308	C6ELS: MOV STATE, #6
		309	;
00AA	2300	310	C6END: MOV A, #0
00AC	2F	311	XCH A, HAMDAT
00AD	93	312	RETR
		313	;
		314	DO; /*CASE 7, WAITING FOR FEED OFF */
		315	TIME(LINESFEED);
		316	IF PFS=0 THEN
		317	DO;
		318	STATE=0;
		319	END;
00AE	23FA	320	CASE7: MOV A, #TLFEED
00B0	62	321	MOV T, A
00B1	55	322	STRT T
00B2	36B8	323	C7ELS
00B4	BE00	324	MOV STATE, #0
00B6	04BA	325	JMP C7END
		326	;
		327	ELSE DO;
		328	STATE=7;
		329	END;
		330	HAMMERSDATA=0;
		331	END; /*END OF CASE 7 */
00B8	BE07	332	C7ELS: MOV STATE, #7
00BA	2300	333	C7END: MOV A, #0
00BC	2F	334	XCH A, HAMDAT
00BD	93	335	RETR
		336	;
		337	END; /* END OF CASE BLOCK */
		338	;
		339	;
		340	;
		341	*****
		342	;
		343	BMGR
		344	;
		345	THIS SEGMENT CONTROLS THE HANDSHAKING BETWEEN THE
		346	CONTROLLER AND THE MASTER PROCESSOR.
		347	;
		348	*****
		349	;
		350	;
		351	;/BMGR-BUFFER MANAGER*/
		352	;
		353	DO;
		354	IF IBF=FULL THEN
		355	DO;
		356	IF TYPE=DATA THEN
		357	DO;
		358	MOTOR=ON;
		359	TEMP=INCQPT(INPNT);
		360	;
		361	;
0100		362	ORG 100H
		363	;
0100	D647	364	BMGR: JNIBF BBPRT
0102	7636	365	JF1 BBCMD
0104	9ABF	366	ANL P2, #MOTON
0106	F8	367	MOV A, INPNT
0107	3470	368	CALL INCQPT

LOC	OBJ	SEQ	SOURCE STATEMENT
		369	IF TEMP<>OUT\$POINT THEN
		370	DO;
		371	IN\$POINT=TEMP;
		372	IF FILL\$MODE=ON THEN
		373	DO;
		374	TEMP=SPACE;
		375	ELSE DO;
		376	TEMP=INPUT\$BUFFER;
		377	INTERRUPT=OFF;
		378	END;
		379	IF TEMP=EOL THEN
		380	DO;
		381	FILL\$MODE=ON;
		382	TEMP=SPACE;
		383	END;
		384	IF TEMP=CONTROL\$CODE THEN TEMP='!';
		385	BUFFER(IN\$POINT)=TEMP AND 03FH;
		386	IF IN\$POINT=BUFFER\$MAX THEN
		387	DO;
		388	FILL\$MODE=OFF;
		389	OUT\$POINT=BUFFER\$MIN;
		390	END;
		391	END;
		392	
		393	
0109	D9	394	BBL1: XRL A,OUTPNT
010A	C647	395	JZ BBPRT
010C	D9	396	XRL A,OUTPNT
010D	A8	397	MOV INPNT,A
010E	0A	398	IN A,P2
010F	3216	399	JB1 FILL
0111	22	400	IN A,DBB
0112	9ADF	401	ANL P2,#NOT(EXREQ)
0114	2418	402	JMP BBL1A
0116	2320	403	FILL: MOV A,SPACE
0118	D30A	404	BBL1A: XRL A,EOL
011A	9620	405	JNZ BBL1B
011C	8A02	406	ORL P2,#FMODE
011E	2428	407	JMP BBL1C
0120	D30A	408	BBL1B: XRL A,EOL
0122	D228	409	JB6 BBL1C
0124	B228	410	JB5 BBL1C
0126	2321	411	MOV A,#EXCLAIM
0128	533F	412	BBL1C: ANL A,#03FH
012A	A0	413	MOV @INPNT,A
012B	F8	414	MOV A,INPNT
012C	D33F	415	XRL A,BMAX
012E	9647	416	JNZ BBPRT
0130	9AFD	417	ANL P2,#NOT FMODE
0132	B918	418	MOV OUTPNT,#BMIN
0134	2447	419	JMP BBPRT
		420	
		421	ELSE DO: /*TYPE IS COMMAND*/
		422	INTERRUPT=OFF;
		423	IF (PORT0 AND 3)=2 THEN INTENA=OFF;
		424	IF (PORT0 AND 3)=3 THEN INTENA=ON;
		425	END;
0136	22	426	BECMD: IN A,DBB
0137	9ADF	427	ANL P2,#NOT(EXREQ)
0139	5303	428	ANL A,#3
013B	323F	429	JB1 BBL2
013D	2447	430	JMP BBPRT
013F	1245	431	BBL2: JB0 BBL3
0141	9AFE	432	ANL P2,#NOT INTENA
0143	2447	433	JMP BBPRT
0145	8A01	434	BBL3: ORL P2,#INTENA
		435	\$ EJECT

LOC	OBJ	SEQ	SOURCE STATEMENT
		436	;
		437	IF OBUF=EMPTY\$FLAG AND (OUT\$POINT<>BMIN OR STATE
		438	DO;
		439	IF OUT\$POINT<=BUFFER\$MAX THEN
		440	DO;
		441	OBUF=BIN(OUT\$POINT);
		442	OUT\$POINT=OUT\$POINT+1;
		443	END;
		444	END;
0147	FB	445	BBPRT: MOV A,OBUF
0148	D3FF	446	XRL A,#EMIFLG
014A	965E	447	JNZ BINT
014C	F9	448	MOV A,OUTPNT
014D	D318	449	XRL A,#QPTMIN
014F	9658	450	JNZ BBPRTA
0151	FE	451	MOV A,STATE
0152	03FD	452	ADD A,#-3
0154	F258	453	JB7 BBPRTA
0156	245E	454	JMP BINT
0158	F9	455	BBPRTA: MOV A,OUTPNT
0159	D25E	456	JB6 BINT
015B	F1	457	MOV A,@OUTPNT
015C	AB	458	MOV OBUF,A
015D	19	459	INC OUTPNT
		460	;
		461	TEMP=INCQPT(INPNT);
		462	IF TEMP<>OUT\$POINT THEN
		463	DO;
		464	IF INTENA=ON AND FMODE=OFF THEN INTERRUPT=ON;
		465	END;
		466	END;
		467	END
015E	F8	468	BINT: MOV A,INPNT
015F	3470	469	CALL INCQPT
0161	D9	470	XRL A,OUTPNT
0162	C600	471	JZ BMGR
		472	BINTA: ;
0164	0A	473	IN A,P2
0165	37	474	CPL A
0166	1200	475	JB0 BMGR
0168	326C	476	SETINT
016A	2400	477	JMP BMGR
016C	8A20	478	SETINT: ORL P2,#EXREQ
016E	2400	479	JMP BMGR
		480	;
		481	;
		482	;
		483	;
		484	;
		485	;
		486	;
		487	;
		488	;
		489	;
		490	;
		491	;
0170	0301	492	INCQPT: ADD A,#1
0172	D275	493	ADJUST
0174	83	494	RET
0175	2318	495	ADJUST: MOV A,#QPTMIN
0177	A7	496	CPL C
0178	83	497	RET
		498	;
		499	;
		500	;
		501	;
		502	;
		503	;
		504	;
		505	;
		506	;
		507	;
		508	;
		509	;
		510	;
0179	BBFF	511	INIT: MOV OBUF,#EMIFLG
017B	B940	512	MOV OUTPNT,#BMAX+1
017D	B83F	513	MOV INPNT,#BMAX
017F	23F0	514	MOV A,#0F0H
0181	3A	515	OUTL P2,A
0182	22	516	IN A,DEB
0183	83	517	RET
		518	;
		519	;
		520	;
		521	;
		522	;
		523	;
		524	\$ EJECT

LOC	OBJ	SEQ	SOURCE STATEMENT
		525	*****
		526	;
		527	;
		528	COLUMN IS CALLED WITH ICNT EOL TO THE CURRENT COLUMN NUMBER
		529	AND PBUF EOL TO THE CHARACTER TO BE CONVERTED.
		530	COLUMN RETURNS THE APPROPRIATE COLUMN OF DATA FROM THE
		531	CHARACTER GENERATOR TABLE. COLUMN IS LOCATED IN PAGE 2
		532	FOLLOWING THE FIRST HALF OF THE TABLE.
		533	*****
		534	;
02E0		535	ORG 2E0H
		536	;
		537	PROCEDURE COLUMN (PRINT\$BUFFER, ICNT);
		538	DO;
		539	FLAG0=NOT PRINT\$BUFFER[5];
		540	PRINT\$BUFFER[5]=0;
		541	TEMP=7*(PBUF+1)-ICNT
		542	IF FLAG0=OFF THEN
		543	DO;
		544	TEMP=MP3 (TEMP);
		545	PRINT\$BUFFER[5]=1;
		546	END;
		547	ELSE DO;
		548	TEMP=MP2 (TEMP);
		549	END;
		550	END;
02E0 FC		551	COLUMN: MOV A, PBUF
02E1 85		552	CLR F0
02E2 B2E5		553	JB5 NOSET
02E4 95		554	CPL F0
02E5 531F		555	NOSET: ANL A, #01FH
02E7 AC		556	MOV PBUF, A
02E8 E7		557	RL A
02E9 E7		558	RL A
02EA E7		559	RL A
02EB 37		560	CPL A
02EC 6C		561	ADD A, PBUF
02ED 6D		562	ADD A, ICNT
02EE 37		563	CPL A
02EF 0307		564	ADD A, #7
02F1 B6F9		565	JF0 PAG2
02F3 E3		566	MOVP3 A, @A
02F4 2C		567	XCH A, PBUF
02F5 4320		568	ORL A, #20H
02F8 2C		569	XCH A, PBUF
02F9 83		570	RET
02FA 83		571	PAG2: MOVP A, @A
		572	RET
		573	;
		574	;
		575	;
		576	*****
		577	;
		578	CHARACTER GENERATOR TABLES.
		579	THE FIRST HALF OF THESE TABLES IS IN PAGE 2. FOLLOWING THIS HALF
		580	IS THE COLUMN SUBROUTINE. THE SECOND HALF OF THE TABLE IS
		581	IN PAGE 3. THE PLACEMENT OF THESE TABLES IS TO TAKE
		582	ADVANTAGE OF THE MCS-41 MOVP AND MOVP3 INSTRUCTIONS.
		583	;
		584	THE CHARACTERS ARE FORMED BY A SEVEN BY SEVEN MATRIX
		585	OF DOTS. EACH DOT POSITION CORRESPONDS
		586	TO ONE HALF THE NORMAL DOT SPACING OF THE LRC PRINTER.
		587	TO PREVENT EXCEEDING THE "BANDWIDTH" OF THE SOLENOIDS
		588	THE CHARACTERS ARE FORMED SO THAT THE SAME SOLENOID IS
		589	NOT ENERGIZED TWICE IN SUCCESSION. CONSTRUCTING THE
		590	TABLE IN THIS MANNER ALLOWS THE FORMATION OF A CHARACTER IN ONLY
		591	4 PRINT COLUMNS SINCE THREE OF THE DOTS WILL APPEAR BETWEEN
		592	NORMAL COLUMN POSITIONS.
		593	;
		594	THE COMMENT FIELD OF THE TABLE SHOWS THE BIT PATTERN OF THE
		595	CHARACTERS. THE SPACING OF THE PRINTER CHARACTERS CAUSES
		596	DISTORTION OF THE CHARACTERS BUT THE PATTERN IS STILL DISCERNABLE.
		597	;
		598	;
		599	*****
600		600	;

LOC	OBJ	SEQ	SOURCE	STATEMENT
		601		
0200		602	ORG	200H
		603		
		604		
0200	0C	605	DB	0CH
0201	22	606	DB	22H
0202	41	607	DB	41H
0203	58	608	DB	58H
0204	01	609	DB	01H
0205	48	610	DB	48H
0206	00	611	DB	00H
		612		
0207	0F	613	DB	0FH
0208	10	614	DB	10H
0209	24	615	DB	24H
020A	40	616	DB	40H
020B	24	617	DB	24H
020C	10	618	DB	10H
020D	0F	619	DB	0FH
		620		
		621		
020E	7F	622	DB	7FH
020F	00	623	DB	00H
0210	49	624	DB	49H
0211	00	625	DB	00H
0212	08	626	DB	08H
0213	55	627	DB	55H
0214	22	628	DB	22H
		629		
0215	3E	630	DB	3EH
0216	41	631	DB	41H
0217	00	632	DB	00H
0218	41	633	DB	41H
0219	00	634	DB	00H
021A	41	635	DB	41H
021B	22	636	DB	22H
		637		
021C	7F	638	DB	7FH
021D	00	639	DB	00H
021E	41	640	DB	41H
021F	00	641	DB	00H
0220	00	642	DB	00H
0221	41	643	DB	41H
0222	3E	644	DB	3EH
		645		
0223	7F	646	DB	7FH
0224	00	647	DB	00H
0225	49	648	DB	49H
0226	00	649	DB	00H
0227	49	650	DB	49H
0228	00	651	DB	00H
0229	41	652	DB	41H
		653		
022A	7F	654	DB	7FH
022B	00	655	DB	00H
022C	48	656	DB	48H
022D	00	657	DB	00H
022E	48	658	DB	48H
022F	00	659	DB	00H
0230	40	660	DB	40H
		661		
0231	3E	662	DB	3EH
0232	41	663	DB	41H
0233	00	664	DB	00H
0234	41	665	DB	41H
0235	04	666	DB	04H
0236	41	667	DB	41H
0237	26	668	DB	26H
		669		
		670	\$	EJECT

LOC	OBJ	SEQ	SOURCE	STATEMENT		
0238	7F	671	DB	7FH	*****	[H]
0239	00	672	DB	00H		
023A	08	673	DB	08H	*	
023B	00	674	DB	00H		
023C	08	675	DB	08H	*	
023D	00	676	DB	00H		
023E	7F	677	DB	7FH	*****	
		678				
023F	00	679	DB	00H		[I]
0240	41	680	DB	41H	* *	
0241	00	681	DB	00H		
0242	7F	682	DB	7FH	*****	
0243	00	683	DB	00H		
0244	41	684	DB	41H	* *	
0245	00	685	DB	00H		
		686				
0246	02	687	DB	02H	*	[J]
0247	01	688	DB	01H	*	
0248	00	689	DB	00H		
0249	01	690	DB	01H	*	
024A	00	691	DB	00H		
024B	01	692	DB	01H	*	
024C	7E	693	DB	7EH	*****	
		694				
024D	7F	695	DB	7FH	*****	[K]
024E	00	696	DB	00H		
024F	04	697	DB	04H	*	
0250	14	698	DB	14H	* *	
0251	22	699	DB	22H	* *	
0252	41	700	DB	41H	* *	
0253	00	701	DB	00H		
		702				
0254	7F	703	DB	7FH	*****	[L]
0255	00	704	DB	00H		
0256	01	705	DB	01H	*	
0257	00	706	DB	00H		
0258	01	707	DB	01H	*	
0259	00	708	DB	00H		
025A	01	709	DB	01H	*	
		710				
025B	7F	711	DB	7FH	*****	[M]
025C	40	712	DB	40H	*	
025D	20	713	DB	20H		
025E	18	714	DB	18H	**	
025F	20	715	DB	20H	*	
0260	40	716	DB	40H		
0261	3F	717	DB	3FH	*****	
		718				
		719				
0262	7F	720	DB	7FH	*****	[N]
0263	20	721	DB	20H	*	
0264	10	722	DB	10H	*	
0265	08	723	DB	08H		
0266	04	724	DB	04H	*	
0267	00	725	DB	00H		
0268	7F	726	DB	7FH	*****	
		727				
0269	3E	728	DB	3EH	*****	[O]
026A	41	729	DB	41H	*	
026B	00	730	DB	00H		
026C	41	731	DB	41H	*	
026D	00	732	DB	00H		
026E	41	733	DB	41H	*	
026F	3E	734	DB	3EH	*****	
		735				
0270	37	736	DB	37H	*** **	[P]
0271	00	737	DB	00H		
0272	48	738	DB	48H	* *	
0273	00	739	DB	00H		
0274	00	740	DB	00H		
0275	48	741	DB	48H	* *	
0276	30	742	DB	30H	**	
		743				
0277	3E	744	DB	3EH	*****	[Q]
0278	41	745	DB	41H	*	
0279	00	746	DB	00H		
027A	40	747	DB	40H	*	
027B	05	748	DB	05H	* *	
027C	42	749	DB	42H	* *	
027D	3D	750	DB	3DH	* *****	

LOC	OBJ	SEQ	SOURCE STATEMENT
		751	
027E	7F	752	DB 7FH ***** ; [R]
027F	00	753	DB 00H
0280	48	754	DB 48H *
0281	00	755	DB 00H *
0282	04	756	DB 04H *
0283	4A	757	DB 4AH *
0284	31	758	DB 31H *
		759	
0285	32	760	DB 32H * * * * ; [S]
0286	49	761	DB 49H *
0287	00	762	DB 00H *
0288	49	763	DB 49H *
0289	00	764	DB 00H *
028A	49	765	DB 49H *
028B	26	766	DB 26H *
		767	
		768	
028C	40	769	DB 40H * ; [T]
028D	00	770	DB 00H
028E	40	771	DB 40H *
028F	3F	772	DB 3FH *****
0290	40	773	DB 40H *
0291	00	774	DB 00H *
0292	40	775	DB 40H *
		776	
0293	7C	777	DB 7CH ***** ; [U]
0294	02	778	DB 02H *
0295	01	779	DB 01H *
0296	00	780	DB 00H *
0297	01	781	DB 01H *
0298	02	782	DB 02H *
0299	7C	783	DB 7CH *****
		784	
029A	78	785	DB 78H ***** ; [V]
029B	04	786	DB 04H *
029C	02	787	DB 02H *
029D	01	788	DB 01H *
029E	02	789	DB 02H *
029F	04	790	DB 04H *
02A0	78	791	DB 78H *****
		792	
02A1	7E	793	DB 7EH ***** ; [W]
02A2	01	794	DB 01H *
02A3	02	795	DB 02H *
02A4	0C	796	DB 0CH *
02A5	02	797	DB 02H *
02A6	01	798	DB 01H *
02A7	7E	799	DB 7EH *****
		800	
02A8	41	801	DB 41H * * * * ; [X]
02A9	22	802	DB 22H *
02AA	14	803	DB 14H *
02AB	08	804	DB 08H *
02AC	14	805	DB 14H *
02AD	22	806	DB 22H *
02AE	41	807	DB 41H *
		808	
02AF	40	809	DB 40H * * * * ; [Y]
02B0	20	810	DB 20H *
02B1	10	811	DB 10H *
02B2	0F	812	DB 0FH *
02B3	10	813	DB 10H *
02B4	20	814	DB 20H *
02B5	40	815	DB 40H *
		816	
		817	\$ EJECT

LOC	OBJ	SEQ	SOURCE STATEMENT
02B6	41	818	DB 41H ; * * *
02B7	02	819	DB 02H ; * * *
02B8	45	820	DB 45H ; * * *
02B9	08	821	DB 08H ; * * *
02BA	51	822	DB 51H ; * * *
02BB	20	823	DB 20H ; * * *
02BC	41	824	DB 41H ; * * *
02BD	7F	825	DB 7FH ; *****
02BE	00	826	DB 00H ; [ ]
02BF	41	827	DB 41H ; * *
02C0	00	828	DB 00H ; * *
02C1	41	829	DB 41H ; * *
02C2	00	830	DB 00H ; * *
02C3	41	831	DB 41H ; * *
02C4	40	832	DB 40H ; * *
02C5	20	833	DB 20H ; * *
02C6	10	834	DB 10H ; * *
02C7	08	835	DB 08H ; * *
02C8	04	836	DB 04H ; * *
02C9	02	837	DB 02H ; * *
02CA	01	838	DB 01H ; * *
02CB	41	839	DB 41H ; * *
02CC	00	840	DB 00H ; [ ]
02CD	41	841	DB 41H ; * *
02CE	00	842	DB 00H ; * *
02CF	41	843	DB 41H ; * *
02D0	00	844	DB 00H ; *****
02D1	7F	845	DB 7FH ; *****
02D2	00	846	DB 00H ; [UA]
02D3	04	847	DB 04H ; * *
02D4	08	848	DB 08H ; * *
02D5	10	849	DB 10H ; * *
02D6	08	850	DB 08H ; * *
02D7	04	851	DB 04H ; * *
02D8	00	852	DB 00H ; [ ]
02D9	01	853	DB 01H ; * *
02DA	00	854	DB 00H ; * *
02DB	01	855	DB 01H ; * *
02DC	00	856	DB 00H ; * *
02DD	01	857	DB 01H ; * *
02DE	00	858	DB 00H ; * *
02DF	01	859	DB 01H ; * *
		860	DB 00H ; *****
		861	DB 00H ; *****
		862	DB 00H ; *****
		863	DB 00H ; *****
		864	DB 00H ; *****
		865	DB 00H ; *****
		866	DB 00H ; *****
		867	DB 00H ; *****
		868	DB 00H ; *****
		869	DB 00H ; *****
		870	DB 00H ; *****
		871	DB 00H ; *****
		872	DB 00H ; *****
		873	DB 00H ; *****
		874	DB 00H ; *****
0300		875	ORG 300H
		876	DB 00H ; [ ]
0300	00	877	DB 00H ; [ ]
0301	00	878	DB 00H ; [ ]
0302	00	879	DB 00H ; [ ]
0303	00	880	DB 00H ; [ ]
0304	00	881	DB 00H ; [ ]
0305	00	882	DB 00H ; [ ]
0306	00	883	DB 00H ; [ ]
		884	DB 00H ; [ ]
		885	DB 00H ; [ ]
0307	00	886	DB 00H ; [ ]
0308	00	887	DB 00H ; [ ]
0309	00	888	DB 00H ; [ ]
030A	7D	889	DB 7DH ; *****
030B	00	890	DB 00H ; *****
030C	00	891	DB 00H ; *****
030D	00	892	DB 00H ; *****
		893	DB 00H ; *****
		894	EJECT

LOC	OBJ	SEQ	SOURCE	STATEMENT	
030E	00	895	DB	00H	; ["
030F	20	896	DB	20H	*
0310	40	897	DB	40H	*
0311	00	898	DB	00H	*
0312	20	899	DB	20H	*
0313	40	900	DB	40H	*
0314	00	901	DB	00H	
		902			
0315	14	903	DB	14H	; [#
0316	00	904	DB	00H	*
0317	7F	905	DB	7FH	*****
0318	00	906	DB	00H	*
0319	7F	907	DB	7FH	*****
031A	00	908	DB	00H	*
031B	14	909	DB	14H	*
		910			
031C	00	911	DB	00H	; [\$
031D	32	912	DB	32H	*
031E	49	913	DB	49H	*
031F	36	914	DB	36H	*
0320	49	915	DB	49H	*
0321	26	916	DB	26H	*
0322	00	917	DB	00H	
		918			
0323	51	919	DB	51H	; [%
0324	02	920	DB	02H	*
0325	54	921	DB	54H	*
0326	08	922	DB	08H	*
0327	15	923	DB	15H	*
0328	20	924	DB	20H	*
0329	45	925	DB	45H	*
		926			
032A	26	927	DB	26H	; [&
032B	49	928	DB	49H	*
032C	10	929	DB	10H	*
032D	49	930	DB	49H	*
032E	26	931	DB	26H	*
032F	01	932	DB	01H	*
0330	05	933	DB	05H	*
		934			
0331	00	935	DB	00H	; ['
0332	00	936	DB	00H	
0333	10	937	DB	10H	*
0334	20	938	DB	20H	*
0335	40	939	DB	40H	*
0336	00	940	DB	00H	
0337	00	941	DB	00H	
		942			
		943			
0338	1C	944	DB	1CH	; [(
0339	22	945	DB	22H	*
033A	41	946	DB	41H	*
033B	00	947	DB	00H	
033C	00	948	DB	00H	
033D	00	949	DB	00H	
033E	00	950	DB	00H	
		951			
033F	00	952	DB	00H	; [)
0340	00	953	DB	00H	
0341	00	954	DB	00H	
0342	00	955	DB	00H	
0343	41	956	DB	41H	*
0344	22	957	DB	22H	*
0345	1C	958	DB	1CH	****
		959			
0346	49	960	DB	49H	; [*
0347	22	961	DB	22H	*
0348	1C	962	DB	1CH	****
0349	77	963	DB	77H	****
034A	1C	964	DB	1CH	****
034B	22	965	DB	22H	*
034C	49	966	DB	49H	*
		967			
034D	08	968	DB	08H	; [+
034E	08	969	DB	08H	*
034F	08	970	DB	08H	*
0350	3E	971	DB	3EH	*****
0351	08	972	DB	08H	*
0352	08	973	DB	08H	*
0353	08	974	DB	08H	*

LOC	OBJ	SEQ	SOURCE STATEMENT
		975	
0354	00	976	DB 00H ; [ , ]
0355	00	977	DB 00H ;
0356	00	978	DB 00H ;
0357	01	979	DB 01H ; **
0358	06	980	DB 06H ;
0359	00	981	DB 00H ;
035A	00	982	DB 00H ;
		983	
035B	04	984	DB 04H ; * ; [-]
035C	04	985	DB 04H ; *
035D	04	986	DB 04H ; *
035E	04	987	DB 04H ; *
035F	04	988	DB 04H ; *
0360	04	989	DB 04H ; *
0361	04	990	DB 04H ;
		991	
		992	
0362	00	993	DB 00H ; [ . ]
0363	00	994	DB 00H ;
0364	00	995	DB 00H ;
0365	01	996	DB 01H ; *
0366	00	997	DB 00H ;
0367	00	998	DB 00H ;
0368	00	999	DB 00H ;
		1000	
0369	01	1001	DB 01H ; * ; [/]
036A	02	1002	DB 02H ; *
036B	04	1003	DB 04H ; *
036C	08	1004	DB 08H ; *
036D	10	1005	DB 10H ; *
036E	20	1006	DB 20H ; *
036F	40	1007	DB 40H ; *
		1008	
0370	1D	1009	DB 1DH ; * * * * ; [0]
0371	22	1010	DB 22H ; * * * *
0372	45	1011	DB 45H ; * * * *
0373	08	1012	DB 08H ; * * * *
0374	51	1013	DB 51H ; * * * *
0375	22	1014	DB 22H ; * * * *
0376	5C	1015	DB 5CH ; * * * *
		1016	
0377	00	1017	DB 00H ; ; [1]
0378	21	1018	DB 21H ; *
0379	40	1019	DB 40H ; *
037A	7F	1020	DB 7FH ; * * * * *
037B	00	1021	DB 00H ;
037C	01	1022	DB 01H ; *
037D	00	1023	DB 00H ;
		1024	
037E	23	1025	DB 23H ; * * * ; [2]
037F	44	1026	DB 44H ; * * *
0380	01	1027	DB 01H ; *
0381	48	1028	DB 48H ; * *
0382	01	1029	DB 01H ; *
0383	48	1030	DB 48H ; * *
0384	31	1031	DB 31H ; * *
		1032	
0385	42	1033	DB 42H ; * * ; [3]
0386	01	1034	DB 01H ; *
0387	50	1035	DB 50H ; * *
0388	01	1036	DB 01H ; *
0389	50	1037	DB 50H ; * *
038A	29	1038	DB 29H ; * * *
038B	46	1039	DB 46H ; * * *
		1040	
		1041	\$ EJECT

LOC	OBJ	SEQ	SOURCE	STATEMENT	
038C	04	1042	DB	04H	; * ; [4]
038D	08	1043	DB	08H	; * ;
038E	14	1044	DB	14H	; * * ;
038F	20	1045	DB	20H	; * * * ;
0390	5F	1046	DB	5FH	; * * * * * ;
0391	00	1047	DB	00H	; ;
0392	04	1048	DB	04H	; * ;
		1049			
0393	72	1050	DB	72H	; * * * * ; [5]
0394	01	1051	DB	01H	; * ;
0395	50	1052	DB	50H	; * * ;
0396	01	1053	DB	01H	; * ;
0397	40	1054	DB	40H	; * * ;
0398	11	1055	DB	11H	; * * * ;
0399	4E	1056	DB	4EH	; * * * * ;
		1057			
039A	17	1058	DB	17H	; * * * * ; [6]
039B	21	1059	DB	21H	; * * * ;
039C	40	1060	DB	40H	; * * * ;
039D	09	1061	DB	09H	; * * * ;
039E	40	1062	DB	40H	; * * * ;
039F	09	1063	DB	09H	; * * * ;
03A0	46	1064	DB	46H	; * * * ;
		1065			
03A1	40	1066	DB	40H	; * ; [7]
03A2	00	1067	DB	00H	; ;
03A3	47	1068	DB	47H	; * * * * ;
03A4	08	1069	DB	08H	; * * * ;
03A5	50	1070	DB	50H	; * * * ;
03A6	20	1071	DB	20H	; * * * ;
03A7	40	1072	DB	40H	; * * * ;
		1073			
03A8	36	1074	DB	36H	; * * * * ; [8]
03A9	49	1075	DB	49H	; * * * * ;
03AA	00	1076	DB	00H	; ;
03AB	49	1077	DB	49H	; * * * * ;
03AC	00	1078	DB	00H	; ;
03AD	49	1079	DB	49H	; * * * * ;
03AE	36	1080	DB	36H	; * * * * ;
		1081			
03AF	30	1082	DB	30H	; * * * ; [9]
03B0	48	1083	DB	48H	; * * * ;
03B1	01	1084	DB	01H	; * * * ;
03B2	48	1085	DB	48H	; * * * ;
03B3	01	1086	DB	01H	; * * * ;
03B4	42	1087	DB	42H	; * * * * ;
03B5	3C	1088	DB	3CH	; * * * * ;
		1089			
		1090			
03B6	00	1091	DB	00H	; ; [:]
03B7	00	1092	DB	00H	; ;
03B8	00	1093	DB	00H	; ;
03B9	14	1094	DB	14H	; * * ;
03BA	00	1095	DB	00H	; ;
03BB	00	1096	DB	00H	; ;
03BC	00	1097	DB	00H	; ;
		1098			
03BD	00	1099	DB	00H	; ; [;]
03BE	00	1100	DB	00H	; ;
03BF	01	1101	DB	01H	; * ;
03C0	02	1102	DB	02H	; * * ;
03C1	14	1103	DB	14H	; * * ;
03C2	00	1104	DB	00H	; ;
03C3	00	1105	DB	00H	; ;
		1106			
03C4	00	1107	DB	00H	; ; [<]
03C5	08	1108	DB	08H	; * * * ;
03C6	14	1109	DB	14H	; * * * * ;
03C7	22	1110	DB	22H	; * * * * ;
03C8	41	1111	DB	41H	; * * * * ;
03C9	00	1112	DB	00H	; ;
03CA	00	1113	DB	00H	; ;
		1114			
03CB	00	1115	DB	00H	; ; [=]
03CC	14	1116	DB	14H	; * * ;
03CD	00	1117	DB	00H	; ;
03CE	14	1118	DB	14H	; * * ;
03CF	00	1119	DB	00H	; ;
03D0	14	1120	DB	14H	; * * ;
03D1	00	1121	DB	00H	; ;

LOC	OBJ	SEQ	SOURCE STATEMENT
		1122	
03D2	00	1123	DB 00H ; [>]
03D3	00	1124	DB 00H ;
03D4	41	1125	DB 41H ;
03D5	22	1126	DB 22H ; *
03D6	14	1127	DB 14H ; *
03D7	08	1128	DB 08H ; *
03D8	00	1129	DB 00H ;
		1130	
03D9	00	1131	DB 00H ; [>]
03DA	20	1132	DB 20H ;
03DB	40	1133	DB 40H ;
03DC	05	1134	DB 05H ; *
03DD	48	1135	DB 48H ; *
03DE	30	1136	DB 30H ; *
03DF	00	1137	DB 00H ;
		1138	
		1139	END

# Using The 8251 Universal Synchronous/Asynchronous Receiver/Transmitter

by Lionel Smith

NEW PRODUCT INFORMATION — 8251A .....	2-33
INTRODUCTION .....	2-34
COMMUNICATION FORMATS .....	2-34
BLOCK DIAGRAM .....	2-35
Receiver .....	2-35
Transmitter .....	2-36
Modem Control .....	2-37
I/O Control .....	2-37
INTERFACE SIGNALS .....	2-37
CPU-Related Signals .....	2-39
Device-Related Signals .....	2-40
MODE SELECTION .....	2-40
PROCESSOR DATA LINK .....	2-43
CONCLUSION .....	2-49
APPENDIX A — 8251 DESIGN HINTS .....	2-63

## NEW PRODUCT INFORMATION

### 8251A

The industry standard USART, the Intel® 8251 has now been improved and is called 8251A. It is packed with features and enhancements as described below. Using the 8251A considerably simplifies programming and minimizes CPU overhead even further.

#### FEATURES AND ENHANCEMENTS

8251A is an advanced design of the industry standard USART, the Intel® 8251. The 8251A operates with an extended range of Intel microprocessors that includes the new 8085 CPU and maintains compatibility with the 8251. Familiarization time is minimal because of compatibility and involves only knowing the additional features and enhancements, and reviewing the AC and DC specifications of the 8251A.

The 8251A incorporates all the key features of the 8251 and has the following additional features and enhancements:

- 8251A has double-buffered data paths with separate I/O registers for control, status, Data In, and Data Out, which considerably simplifies control programming and minimizes CPU overhead.
- In asynchronous operations, the Receiver detects and handles "break" automatically, relieving the CPU of this task.
- A refined Rx initialization prevents the Receiver from starting when in "break" state, preventing unwanted interrupts from a disconnected USART.
- At the conclusion of a transmission, TxD line will always return to the marking state unless SBRK is programmed.

- Tx Enable logic enhancement prevents a Tx Disable command from halting transmission until all data previously written has been transmitted. The logic also prevents the transmitter from turning off in the middle of a word.
- When External Sync Detect is programmed, Internal Sync Detect is disabled, and an External Sync Detect status is provided via a flip-flop which clears itself upon a status read.
- Possibility of false sync detect is minimized by ensuring that if double character sync is programmed, the characters be contiguously detected and also by clearing the Rx register to all ones whenever Enter Hunt command is issued in Sync mode.
- As long as the 8251A is not selected, the RD and WR do not affect the internal operation of the device.
- The 8251A Status can be read at any time but the status update will be inhibited during status read.
- The 8251A is free from extraneous glitches and has enhanced AC and DC characteristics, providing higher speed and better operating margins.
- Baud rate from DC to 64K.
- Fully compatible with Intel's new industry standard, the MCS-85.

## INTRODUCTION

The Intel 8251 is a Universal Synchronous/Asynchronous Receiver/Transmitter (USART) which is capable of operating with a wide variety of serial communication formats. Since many peripheral devices are available with serial interfaces, the 8251 can be used to interface a microcomputer to a broad spectrum of peripherals, as well as to a serial communications channel. The 8251 is part of the MCS-80™ Microprocessor Family, and as such it is capable of interfacing to the 8080 system with a minimum of external hardware.

This application note describes the 8251 as a component and then explains its use in sample applications via several examples. A specific use of the 8251 to facilitate communication between two MCS-80 systems is discussed in detail from both the hardware and software viewpoints. The first two sections of this application note describe the 8251 first from a functional standpoint and then on a detailed level. The function of each input and output pin is fully defined. The next section describes the various operating modes and how they can be selected, and finally, a sample design is discussed using the 8251 as a data link between the MCS-80 systems.

## COMMUNICATION FORMATS

Serial communications, either on a data link or with a local peripheral, occurs in one of two basic formats; asynchronous or synchronous. These formats are similar in that they both require framing information to be added to the data to enable proper detection of the character at the receiving end. The major difference between the two formats is that the asynchronous format requires framing information to be added to each character, while the synchronous format adds framing information to blocks of data, or messages. Since the synchronous format is more efficient than the asynchronous format but requires more complex decoding, it is typically found on high-speed data links, while the asynchronous format is used on lower speed lines.

The asynchronous format starts with the basic data bits to be transmitted and adds a "START" bit to the front of them and one or more "STOP" bits behind them as they are transmitted. The START bit is a logical zero, or SPACE, and is defined as the positive voltage level by RS-232-C. The STOP bit is a logical one, or MARK, and is defined as the negative voltage level by RS-232-C. In current loop applications current flow normally indicates a MARK and lack of current a SPACE. The START bit tells the receiver to start assembling a character and allows the receiver to synchronize itself with the transmitter. Since this synchronization only

has to last for the duration of the character (the next character will contain a new START bit), this method works quite well assuming a properly designed receiver. One or more STOP bits are added to the end of the character to ensure that the START bit of the next character will cause a transition on the communication line and to give the receiver time to "catch up" with the transmitter if its basic clock happens to be running slightly slower than that of the transmitter. If, on the other hand, the receiver clock happens to be running slightly faster than the transmitter clock, the receiver will perceive gaps between characters but will still correctly decode the data. Because of this tolerance to minor frequency deviations, it is not necessary that the transmitter and receiver clocks be locked to the identical frequency for successful asynchronous communication.

The synchronous format, instead of adding bits to each character, groups characters into records and adds framing characters to the record. The framing characters are generally known as SYN characters and are used by the receiver to determine where the character boundaries are in a string of bits. Since synchronization must be held over a fairly long stream of data, bit synchronization is normally either extracted from the communication channel by the modem or supplied from an external source.

An example of the synchronous and asynchronous formats is shown in Figure 1. The synchronous format shown is fairly typical in that it requires two SYN characters at the start of the message. The asynchronous format, also typical, requires a START bit preceding each character and a single STOP bit following it. In both cases, two 8-bit characters are to be transmitted. In the asynchronous mode  $10 \times n$  bits are used to transmit  $n$  characters and in the synchronous mode  $8N + 16$  bits are used. For the example shown the asynchronous mode is actually more efficient, using 20 bits versus 32. To transmit a thousand characters in the asynchronous mode, however, takes 10,000 bits versus 8,016 for the synchronous format mode. For long messages the synchronous format becomes much more efficient than the asynchronous format; the crossover point for the examples shown in Figure 1 is eight characters, for which both formats require 80 bits.

In addition to the differences in format between synchronous and asynchronous communication, there are differences with regards to the type of modems that can be used. Asynchronous modems typically employ FSK (Frequency Shift Keying) techniques which simply generate one audio tone for a MARK and another for a SPACE. The receiving modem detects these tones on the telephone

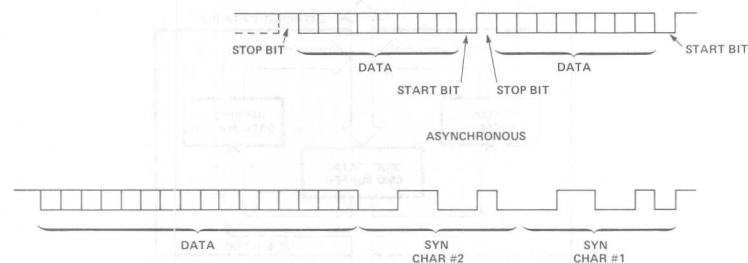


Figure 1. Transmission Formats

line, converts them to logical signals, and presents them to the receiving terminal. Since the modem itself is not concerned with the transmission speed, it can handle baud rates from zero to its maximum speed. Synchronous modems, in contrast to asynchronous modems, supply timing information to the terminal and require data to be presented to them in synchronism with this timing information. Synchronous modems, because of this extra clocking, are only capable of operating at certain preset baud rates. The receiving modem, which has an oscillator running at the same frequency as the transmitting modem, phase locks its clock to that of the transmitter and interprets changes of phase as data.

In some cases it is desirable to operate in a hybrid mode which involves transmitting data with the asynchronous format using a synchronous modem. This occurs when an increase in operating speed is required without a change in the basic protocol of the system. This hybrid technique is known as isosynchronous and involves the generation of the start and stop bits associated with the asynchronous format, while still using the modem clock for bit synchronization.

The 8251 USART has been designed to meet a broad spectrum of requirements in the synchronous, asynchronous, and isosynchronous modes. In the synchronous mode the 8251 operates with 5, 6, 7, or 8-bit characters. Even or odd parity can be optionally appended and checked. Synchronization can be achieved either externally via added hardware or internally via SYN character detection. SYN detection can be based on one or two characters which may or may not be the same. The single or double SYN characters are inserted into the data stream automatically if the software fails to supply data in time. The automatic generation of SYN characters is required to prevent the loss of synchronization. In the asynchronous mode the 8251 operates with the same data and parity structures as it does in the synchronous mode. In addition to appending a START bit to this data, the

8251 appends 1, 1½, or 2 STOP bits. Proper framing is checked by the receiver and a status flag set if an error occurs. In the asynchronous mode the USART can be programmed to accept clock rates of 16 or 64 times the required baud rate. Isosynchronous operation is a special case of asynchronous with the multiplier rate programmed as one instead of 16 or 64. Note that X1 operation is only valid if the clocks of the receiver and transmitter are synchronized.

The 8251 USART can transmit the three formats in half or full duplex mode and is double-buffered internally (i.e., the software has a complete character time to respond to a service request). Although the 8251 supports basic data set control signals (e.g., DTR and RTS), it does not fully support the signaling described in EIA-RS-232-C. Examples of unsupported signals are Carrier Detect (CF), Ring Indicator (CE), and the secondary channel signals. In some cases an additional port will be required to implement these signals. The 8251 also does not interface to the voltage levels required by EIA-RS-232-C; drivers and receivers must be added to accomplish this interface.

## BLOCK DIAGRAM

A block diagram of the 8251 is shown in Figure 2. As can be seen in the figure, the 8251 consists of five major sections which communicate with each other on an internal data bus. The five sections are the receiver, transmitter, modem control, read/write control, and I/O Buffer. In order to facilitate discussion, the I/O Buffer has been shown broken down into its three major subsections: the status buffer, the transmit data/command buffer, and the receive data buffer.

## Receiver

The receiver accepts serial data on the RxD pin and converts it to parallel data according to the appropriate format. When the 8251 is in the asynchronous mode and it is ready to accept a character

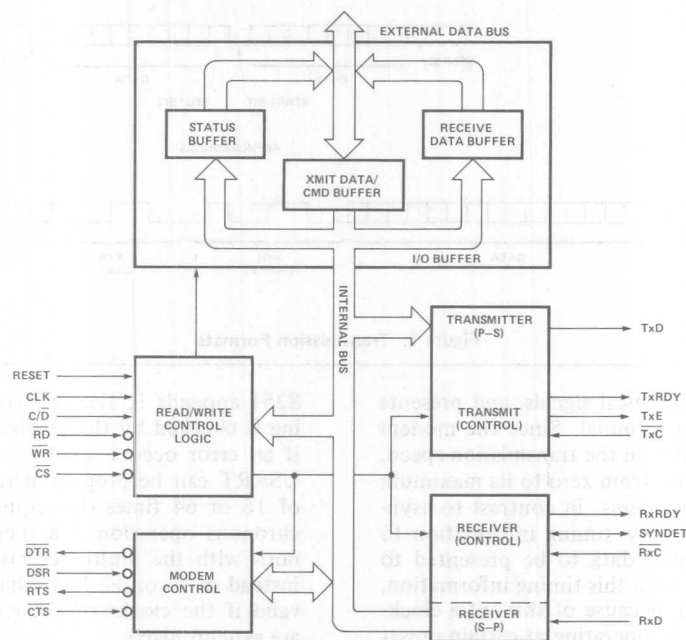


Figure 2. 8251 Block Diagram

(i.e., it is not in the process of receiving a character), it looks for a low level on the Rx D line. When it sees the low level, it assumes that it is a START bit and enables an internal counter. At a count equivalent to one-half of a bit time, the Rx D line is sampled again. If the line is still low, a valid START bit has probably been received and the 8251 proceeds to assemble the character. If the Rx D line is high when it is sampled, then either a noise pulse has occurred on the line or the receiver has become enabled in the middle of the transmission of a character. In either case the receiver aborts its operation and prepares itself to accept a new character. After the successful reception of a START bit the 8251 clocks in the data, parity, and STOP bits, and then transfers the data on the internal data bus to the receive data register. When operating with less than 8 bits, the characters are right-justified. The RxRDY signal is asserted to indicate that a character is available.

In the synchronous mode the receiver simply clocks in the specified number of data bits and transfers them to the receiver buffer register, setting RxRDY. Since the receiver blindly groups data bits into characters, there must be a means of synchronizing the receiver to the transmitter so that the proper character boundaries are maintained in the serial data stream. This synchronization is achieved in the HUNT mode.

In the HUNT mode the 8251 shifts in data on the

Rx D line one bit at a time. After each bit is received, the receiver register is compared to a register holding the SYN character (program loaded). If the two registers are not equal, the 8251 shifts in another bit and repeats the comparison. When the registers compare as equal, the 8251 ends the HUNT mode and raises the SYNDET line to indicate that it has achieved synchronization. If the USART has been programmed to operate with two SYN characters the process is as described above, except that two contiguous characters from the line must compare to the two stored SYN characters before synchronization is declared. Parity is not checked. If the USART has been programmed to accept external synchronization, the SYNDET pin is used as an input to synchronize the receiver. The timing necessary to do this is discussed in the SIGNALS section of this note. The USART enters the HUNT mode when it is initialized into the synchronous mode or when it is commanded to do so by the command instruction. Before the receiver is operated, it must be enabled by the Rx E bit (D<sub>2</sub>) of the command instructions. If this bit is not set the receiver will not assert the RxRDY bit.

#### Transmitter

The transmitter accepts parallel data from the processor, adds the appropriate framing information, serializes it, and transmits it on the Tx D pin. In the asynchronous mode the transmitter always

adds a START bit; depending on how the unit is programmed, it also adds an optional even or odd parity bit, and either 1, 1½, or 2 STOP bits. In the synchronous mode no extra bits (other than parity, if enable) are generated by the transmitter unless the computer fails to send a character to the USART. If the USART is ready to transmit a character and a new character has not been supplied by the computer, the USART will transmit a SYN character. This is necessary since synchronous communications, unlike asynchronous communications, does not allow gaps between characters. If the USART is operating in the dual SYN mode, both SYN characters will be transmitted before the message can be resumed. The USART will not generate SYN characters until the software has supplied at least one character; i.e., the USART will fill 'holes' in the transmission but will not initiate transmission itself. The SYN characters which are to be transmitted by the USART are specified by the software during the initialization procedure. In either the synchronous or asynchronous modes, transmission is inhibited until TxEnable and the CTS input are asserted.

An additional feature of the transmitter is the ability to transmit a BREAK. A BREAK is a period of continuous SPACE on the communication line and is used in full duplex communication to interrupt the transmitting terminal. The 8251 USART will transmit a BREAK condition as long as bit 3 (SBRK) of the command register is set.

### Modem Control

The modem control section provides for the generation of RTS and the reception of CTS. In addition, a general purpose output and a general purpose input are provided. The output is labeled DTR and the input is labeled DSR. DTR can be asserted by setting bit 2 of the command instruction; DSR can be sensed as bit 7 of the status register. Although the USART itself attaches no special significance to these signals, DTR (Data Terminal Ready) is normally assigned to the modem, indicating that the terminal is ready to communicate and DSR (Data Set Ready) is a signal from the modem indicating that it is ready for communications.

### I/O Control

The Read/Write Control Logic decodes control signals on the 8080 control bus into signals which gate data on and off the USART's internal bus and controls the external I/O bus (DB<sub>0</sub>-DB<sub>7</sub>). The truth table for these operations is as follows:

If neither READ or WRITE is a zero, then the USART will not perform an I/O function. READ

CE	C/D	READ	WRITE	Function
0	0	0	1	CPU Reads Data from USART
0	1	0	1	CPU Reads Status from USART
0	0	1	0	CPU Writes Data to USART
0	1	1	0	CPU Writes Command to USART
1	X	X	X	USART Bus Floating (NO-OP)

and WRITE being a zero at the same time is an illegal state with undefined results. The Read/Write Control Logic contains synchronization circuits so that the READ and WRITE pulses can occur at any time with respect to the clock inputs to the USART.

The I/O buffer contains the STATUS buffer, the RECEIVE DATA buffer and the XMIT DATA/CMD buffer as shown in Figure 2. Note that although there are two registers which store data for transfer to the CPU (STATUS and RECEIVE DATA), there is only one register which stores data being transferred to the USART. The sharing of the input register for both transmit data and commands makes it important to ensure that the USART does not have data stored in this register before sending a command to the device. The TxRDY signal can be monitored to accomplish this. Neither data nor commands should be transferred to the USART if TxRDY is low. Failure to perform this check can result in erroneous data being transmitted.

### INTERFACE SIGNALS

The interface signals of the 8251 USART can be broken down into two groups — a CPU-related group and a device-related group. The CPU-related signals have been designed to optimize the attachment of the 8251 to a MCS-80™ system. The device-related signals are intended to interface a modem or like device. Since many peripherals (TTY, CRT, etc.) can be obtained with a modem-like interface, the USART has a broad range of applications which do not include a modem. Note that although the USART provides a logical interface to an EIA-RS-232 device, it does not provide EIA compatible drive, and this must be added via circuitry external to the 8251. As an example of a peripheral interface application and to aid in understanding the signal descriptions which follow, Figure 3 shows a system configured to interface with a TTY or CRT.

CONNECT APPROPRIATE JUMPER PAIRS FOR APPROPRIATE USE

BAND DATA TABLE			
CHT	TTY	TTL	PAD 20 TO BAND DATA X04
23 TO 24	23 TO 26	23 TO 25	31
18 TO 19	18 TO 19	18 TO 17	4800
14 TO 15	14 TO 15	14 TO 13	1200
10 TO 11	10 TO 11	10 TO 10	34
6 TO 8	7 TO 8	7 TO 8	300
2 TO 3	2 TO 3	2 TO 3	35
23 TO 21	23 TO 21	23 TO 21	70 TO 100 K TO 5 IS CONNECTED

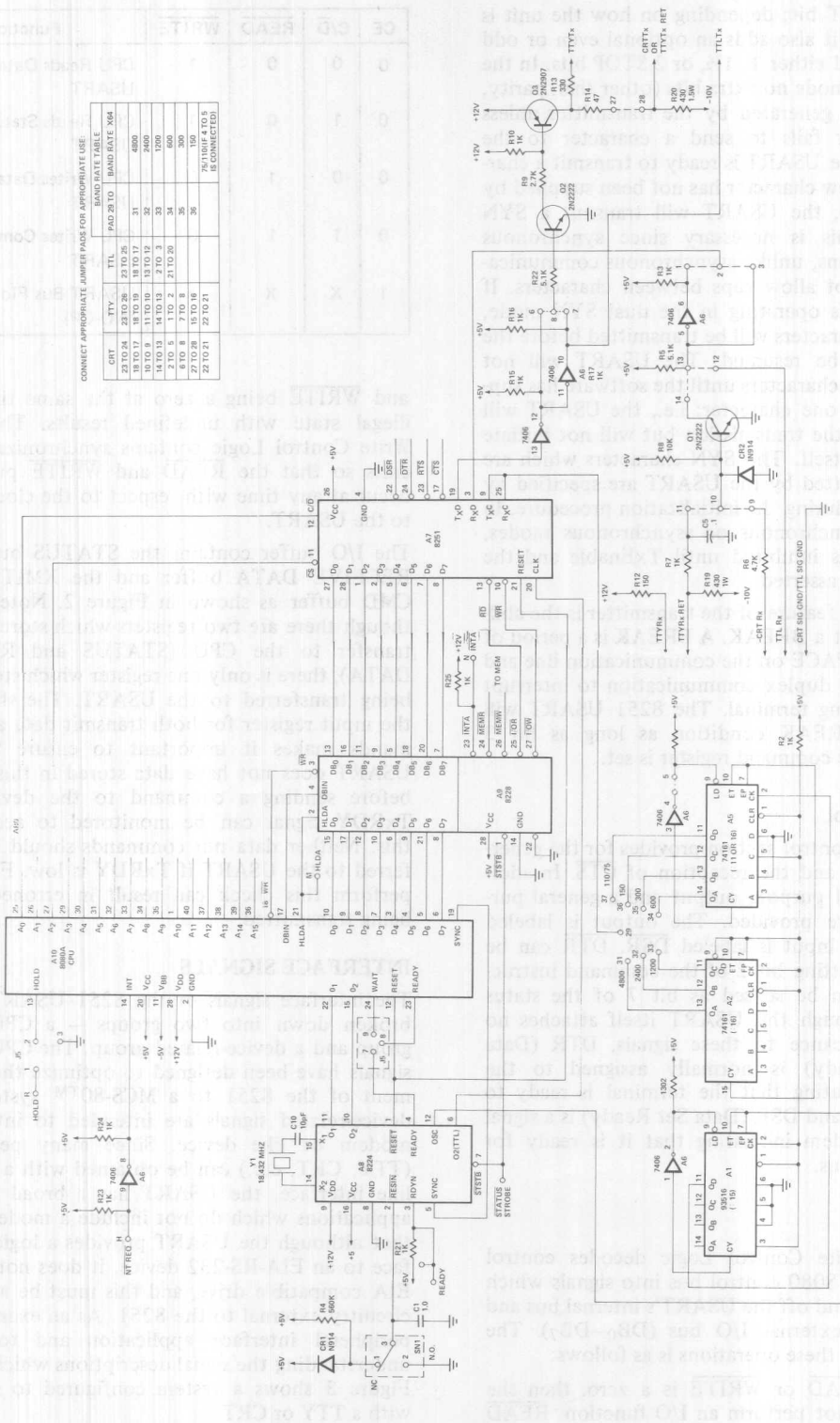


Figure 3. Terminal Interface

# CPU-Related Signals

V <sub>CC</sub> (26)	I	+5 Volt Supply
GND (4)	I	+5 Volt Common
CLK (20)	I	The CLK input generates internal device timing. No external inputs or outputs are referenced to CLK, but the frequency of CLK must be greater than 30 times the Receiver or Transmitter clock inputs for synchronous mode or 4.5 times the clock inputs for an asynchronous mode. An additional constraint is imposed by the electrical specifications (ref. Appendix B) which require the period of CLK be between 0.42 $\mu$ sec and 1.35 $\mu$ sec. The CLK input can generally be connected to the Phase 2 (TTL) output of the 8224 clock generator.
RESET (21)	I	A high on this input performs a master reset on the 8251. The device returns to the idle mode and will remain there until reinitialized with the appropriate control words.
DB <sub>7</sub> -DB <sub>0</sub> (8,7,6,5,2,1,28,27)	I/O	The DB signals form a three-state bus which can be connected to the CPU data bus. Control, status, and data are transferred on this bus. Note that the CPU always remains in control of the bus and all transfers are initiated by it.
$\overline{\text{CS}}$ (11)	I	<i>Chip Select.</i> A low on this input enables communication between the USART and the CPU. Chip Select should go low when the USART is being addressed by the CPU.
C/ $\overline{\text{D}}$ (12)	I	<i>Control/Data.</i> During a read operation this pin selects either status or data to be input to the CPU (high=status, low=data). During a write operation this pin causes the USART to interpret the data on the bus as a command if it is high or as data if it is low.
$\overline{\text{RD}}$ (13)	I	A low on this input causes the USART to gate either

status or data onto the data bus.

**$\overline{\text{WR}}$  (10)** I A low on this input causes the USART to accept data on the data bus as either a command or as a data character.

**TxRDY (15)** O *Transmitter Ready.* This output signals the CPU that the USART is ready to accept a data character or command. It can be used as an interrupt to the system or, for polled operation, the CPU can check TxRDY using the status read operation. Note, however, that while the TxRDY status bit will be asserted whenever the XMIT DATA/CMD buffer is empty, the TxRDY output will be asserted only if the buffer is empty and the USART is enabled to transmit (i.e.,  $\overline{\text{CTS}}$  is low and TxEN is high). TxRDY will be reset when the USART receives a character from the program.

**TxE (18)** O *Transmitter Empty.* A high output on this line indicates that the parallel to serial converter in the transmitter is empty. In the synchronous mode, if the CPU has failed to load a new character in time, TxE will go high momentarily as SYN characters are loaded into the transmitter to fill the gap in transmission.

**RxRDY (14)** O *Receiver Ready.* This output goes high to indicate that the 8251 has received a character on its serial input and is ready to transfer it to the CPU. Although the receiver runs continuously, RxRDY will only be asserted if the RxEN (Receive Enable) bit in the command register has been set. RxRDY can be connected to the interrupt structure or, for polled operation, the CPU can check the condition of RxRDY using a status read operation. RxRDY will be reset when the character is read by the CPU.

**SYNDET (16) I/O** *Synch Detect*. This line is used in the synchronous mode only. It can be either an input or output, depending on whether the initialization program sets the USART for external or internal synchronization. SYNDET is reset to a zero by RESET. When in the internal synchronization mode, the USART uses SYNDET as an output to indicate that the device has detected the required SYN character(s). A high output indicates synchronization has been achieved. If the USART is programmed to operate with double SYN characters, SYNDET will go high in the middle of the last bit of the second SYN character. SYNDET will be reset by a status read operation. When in the external synchronization mode a positive-going input on the SYNDET line will cause the 8251 to start assembling characters on the next falling edge of  $RxC$ . The high input should be maintained at least for one  $RxC$  cycle following this edge.

#### Device-Related Signals

**$\overline{DTR}$  (24) O** *Data Terminal Ready*. This is a general purpose output signal which can be set low by programming a '1' in command instruction bit 1. This signal allows additional device control.

**$\overline{DSR}$  (22) I** *Data Set Ready*. This is a general purpose input signal. The status of this signal can be tested by the CPU through a status read. This pin can be used to test device status and is read as bit 7 of the status register.

**$\overline{RTS}$  (23) O** *Request to Send*. This is a general purpose output signal equivalent to  $\overline{DTR}$ . RTS is normally used to request that the modem prepare itself to transmit (i.e., establish carrier).  $\overline{RTS}$  can be asserted

(brought low) by setting bit 5 in the command instruction.

**$\overline{CTS}$  (17) I** *Clear to Send*. A low on this input enables the USART to transmit data. CTS is normally generated by the modem in response to a  $\overline{RTS}$ .

**$\overline{RxC}$  (25) I** *Receiver Clock*. This clock controls the data rate of characters to be received by the USART. In the synchronous mode  $RxC$  is equivalent to the baud rate, and is supplied by the modem. In asynchronous mode  $\overline{RxC}$  is 1, 16, or 64 times the baud rate. The clock division is preselected by the mode control instruction. Data is sampled by the USART on the rising edge of  $\overline{RxC}$ .

**$RxD$  (3) I** *Receiver Data*. Characters are received serially on this pin and assembled into parallel characters.  $RxD$  is high true (i.e., High = MARK or ONE).

**$\overline{TxC}$  (9) I** *Transmitter Clock*. This clock controls the rate at which characters are transmitted by the USART. The relationship between clock rate and baud rate is the same as for  $RxC$ . Data is shifted out of the USART on the falling edge of  $\overline{TxC}$ .

**$TxD$  (19) O** *Transmit Data*. Parallel characters sent by the CPU are transmitted serially by the USART on this line.  $TxD$  is high true (i.e., High = MARK or ONE).

#### MODE SELECTION

The 8251 USART is capable of operating in a number of modes (e.g., synchronous or asynchronous). In order to keep the hardware as flexible as possible (both at the chip and end product level), these operating modes are selected via a series of control outputs to the USART. These mode control outputs must occur between the time the USART is reset and the time it is utilized for data transfer. Since the USART needs this information to structure its internal logic it is essential to complete the initialization before any attempts are made at data transfer (including reading status).

A flowchart of the initialization process appears in Figure 4. The first operation which must occur following a reset is the loading of the mode control

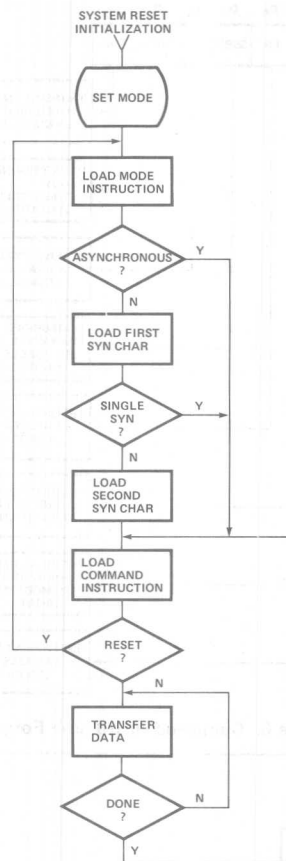


Figure 4. Initialization Flowchart

register. The mode control register is loaded by the first control output ( $C/\overline{D}=1$ ,  $\overline{RD}=1$ ,  $\overline{WR}=0$ ,  $\overline{CS}=0$ ) following a reset. The format of the mode control instruction is shown in Figure 5. The instruction can be considered as four 2-bit fields. The first 2-bit field ( $D_1 D_0$ ) determines whether the USART is to operate in the synchronous (00) or asynchronous mode. In the asynchronous mode this field also controls the clock scaling factor. As an example, if  $D_1$  and  $D_0$  are both ones, the  $\overline{RxC}$  and  $\overline{TxC}$  will be divided by 64 to establish the baud rate. The second field,  $D_3-D_2$ , determines the number of data bits in the character and the third,  $D_5-D_4$ , controls parity generation. Note that the parity bit (if enabled) is added to the data bits and is not considered as part of them when setting up the character length. As an example, standard ASCII transmission, which is seven data bits plus even parity, would be specified as:

X X 1 1 1 0 X X

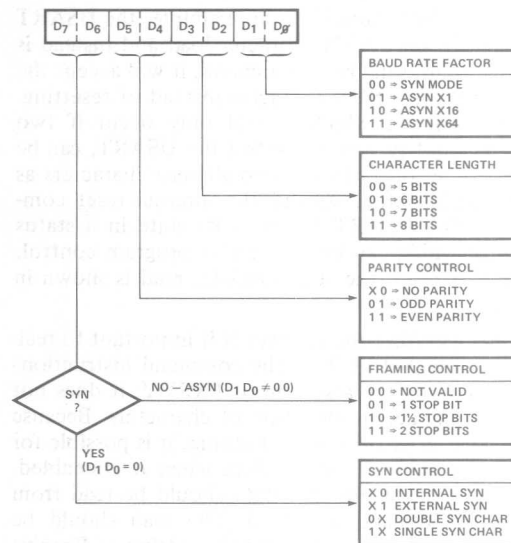


Figure 5. Mode Instruction Format

The last field,  $D_7-D_6$ , has two meanings, depending on whether operation is to be in the synchronous or asynchronous mode. For the asynchronous mode (i.e.,  $D_1 D_0 \neq 00$ ), it controls the number of STOP bits to be transmitted with the character. Since the receiver will always operate with only one STOP bit,  $D_7$  and  $D_6$  only control the transmitter. In the synchronous mode ( $D_1 D_0 = 00$ ), this field controls the synchronizing process. Note that the choice of single or double SYN characters is independent of the choice of internal or external synchronization. This is because even though the receiver may operate with external synchronization logic, the transmitter must still know whether to send one or two SYN characters should the CPU fail to supply a character in time.

Following the loading of the mode instruction the appropriate SYN character (or characters) must be loaded if synchronous mode has been specified. The SYN character(s) are loaded by the same control output instruction used to load the mode instruction. The USART determines from the mode instruction whether no, one, or two SYN characters are required and uses the control output to load SYN characters until the required number are loaded.

At completion of the load of SYN characters (or after the mode instruction in the asynchronous mode), a command character is issued to the USART. The command instruction controls the operation of the USART within the basic framework established by the mode instruction. The format of the command instruction is shown in

Figure 6. Note that if, as an example, the USART is waiting for a SYN character load and instead is issued an internal reset command, it will accept the command as a SYN character instead of resetting. This situation, which should only occur if two independent programs control the USART, can be avoided by outputting three all zero characters as commands before issuing the internal reset command. The USART indicates its state in a status register which can be read under program control. The format of the status register read is shown in Figure 7.

When operating the receiver it is important to realize that Rx E (bit 2 of the command instruction) only inhibits the assertion of RxRDY; it does not inhibit the actual reception of characters. Because the receiver is constantly running, it is possible for it to contain extraneous data when it is enabled. To avoid problems this data should be read from the USART and discarded. The read should be done immediately following the setting of Receive Enable in the asynchronous mode, and following the setting of Enter Hunt in the synchronous mode. It is not necessary to wait for RxRDY before executing the dummy read.

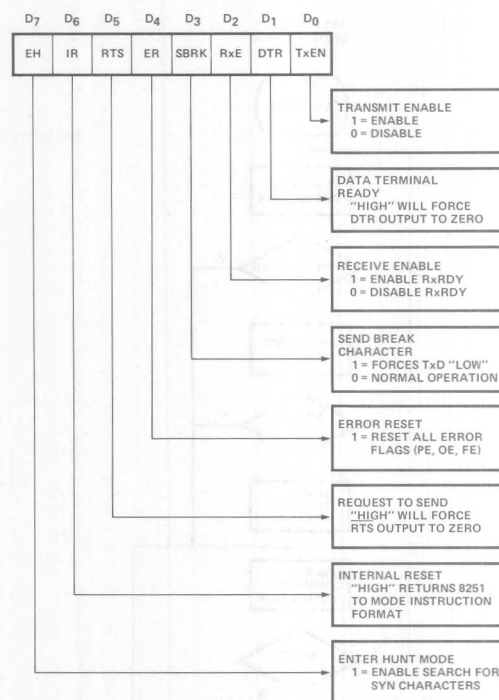


Figure 6. Command Instruction Format

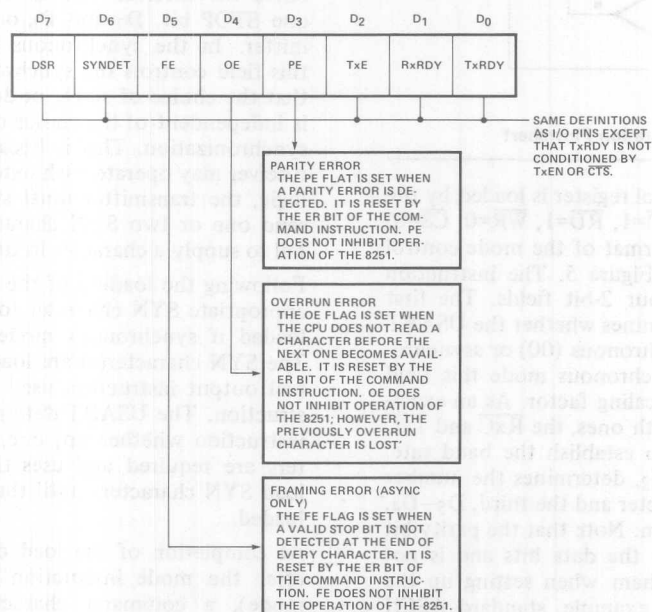


Figure 7. Status Register Format

## PROCESSOR DATA LINK

The ability to change the operating mode of the USART by software makes the 8251 an ideal device to use to implement a serial communication link. A terminal initially configured with a simple asynchronous protocol can be upgraded to a synchronous protocol such as IBM Binary Synchronous Communication by a software only upgrade. In order to demonstrate the use of the 8251 USART, the remainder of this document will describe the implementation of an interrupt-driven, full duplex communication link on the Intel MDS™ system. With minor modifications, the program developed could be used on the Intel SBC-80/10™ OEM card, thus implementing a data link between the two systems. Such a facility can be used to down-load programs, run diagnostics, and maintain common data bases in multiprocessor systems.

The factors which must be considered in the design of such a link include the desired transmission rate and format, the error checking requirements, the desirability of full duplex operation, and the physical implementation of the link. The basic requirement of the system described here is that it allow an Intel SBC-80/10 OEM card to be loaded from an MDS development system, either locally or on the switched telephone network. An additional constraint is that the modem used on the switched network be readily available and inexpensive. These requirements led to the choice of a modem such as the Bell 103A to implement the link. These modems, which support full duplex communication at up to 300 baud, are readily available from a number of sources at reasonable cost. These modems are also available in acoustically coupled versions which do not require permanent installation on the telephone network. Interface to the 103A modem is accomplished with nine wires: Protective Ground, Signal Ground, Transmitted Data, Received Data, Clear to Send, Data Set Ready, Data Terminal Ready, Carrier Detector, and Ringing Indicator.

The utilization of the interface signals to the modem is as follows:

Protective Ground*	Protective Ground is used to bond the chassis ground of the modem to that of the terminal.
Signal Ground	Signal Ground provides a common ground reference between the modem and the terminal.
Transmitted Data	Transmitted Data is used to transfer serial data from the terminal to the modem.

Received Data	Received Data is used to transfer serial data from the modem to the terminal.
Clear to Send	Clear to Send indicates that the modem has established a connection with a remote modem and is ready to transmit data.
Data Set Ready	Data Set Ready indicates that the modem is connected to the telephone line and is in the data mode.
Data Terminal Ready	Data Terminal Ready is a signal from the terminal which permits the modem to enter the data mode.
Carrier Detector	Carrier Detector is identical to Clear to Send in the 103 modem and will not be used in this interface.
Ringing Indicator	Ringing Indicator indicates that the modem is receiving a ringing signal from the telephone system. This signal will not be used in the interface, since it is possible for the terminal to assert Data Terminal Ready whenever it is ready for the modem to "answer the telephone". The modem uses Data Set Ready to indicate that it has answered the call.

A block diagram showing the connections between the MDS and the SBC-80/10 through the modems is shown in Figure 8. Figure 9 shows the portion of the MDS monitor board devoted to the USARTs and Figure 10 shows the equivalent section of the SBC-80/10 board. Note that several signals on the MDS do not have the proper EIA defined voltage levels, and for this reason the adapter shown in Figure 11 was added to the MDS. The 390 pF capacitor was added to the 1488 driver to bring the rise time within EIA imposed limits of 30 volts/μsec. In Figure 7 the signal labels within the MDS and SBC-80/10 blocks correspond to the labels on the schematics, the signal labels within the modem blocks correspond to EIA conventions, and the signal labels on the wires between the blocks are abbreviations for the English language names of the signals.

As an example of how the USART clocks can be generated, circuits A27, A16, and A15 of Figure 9 form a divider of the OSC signal. The OSC signal has a frequency of 18.432 MHz and is generated by the 8224 which generates system timing for the 8080A. The 18.432 MHz signal results in a state time of 488 ns versus the normal 500 ns for the 8080A. (This does not violate 8080A specifications.) The 18.432 MHz signal can be divided by

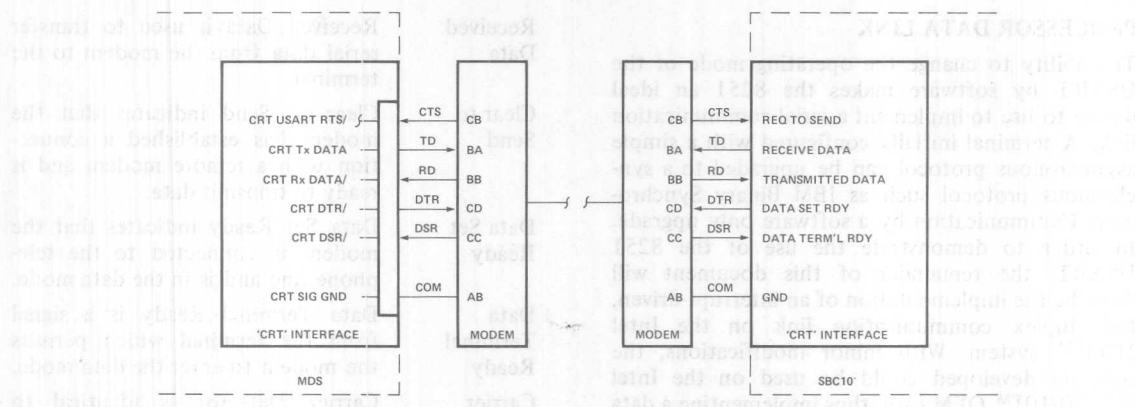


Figure 8. System Block Diagram

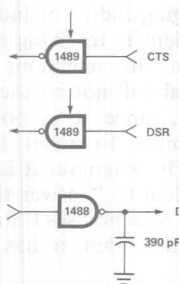


Figure 9. EIA Adapter

30 and then 64 to give a 9600 baud communication standard. The 9600 baud signal can be further divided to give 4800, 2400, 1200, 600, and 300 baud signals. The 1200 baud signal can be divided by 11 to give a 109.1 baud signal which is within 1% of the 110 baud standard signal rate. Note that because of constraints on the CLK input 9600 baud operation is not possible in the X64 mode. The divide by 64 can be accomplished by dividing by 4 with a counter and then 16 within the USART.

In order to keep the system as general purpose as possible, it was decided to transmit 8-bit data characters with an appended odd parity bit. Having a full 8-bit byte available for data enables the transmission of codes such as ASCII (which is 7-level with an additional parity bit) to be transmitted and received transparently in the system. Also, of course, it allows 8-bit bytes from the 8080A memory to be transferred in one transmission character. If error checking beyond the parity check is required, it could be added to the data record to be transmitted in the form of redundant check characters.

Before the software design of the system could be undertaken, it was necessary to decide whether service requests from the USART would be handled on a polled or interrupt driven mode. Polled operation normally results in more compact code but it requires that whatever programs are running concurrently with a transmission or reception must periodically either check the status of the USART or call a routine that does. Since it was not possible to determine what program might be running during a receive or transmit operation, it was decided to operate in an interrupt driven mode.

The program which operates the 8251 must be instructed as to what data it should transmit or receive from some other program resident in the 8080 system. To facilitate the discussion of the operation of the software, the following definitions will be made:

USRUN is the program which controls the operation of the 8251.

USER is a program which utilizes USRUN in order to effect a data transmission.

USER passes commands and parameters to USRUN by means of the control block shown in Figure 12. The first byte of the block contains the command which USER wants USRUN to execute. Valid contents of this byte are "C" which causes USRUN to initialize itself and the 8251, "R" which causes the execution of the data input (or READ) operation, and "W" which causes a data output (WRITE) operation. The second byte of the control block is used by USRUN to inform USER of the status of the requested operation. The third and fourth bytes specify the starting address of a buffer set up by USER which contains the data for a transmit operation or which will be used by USRUN to store received data. The fifth and sixth bytes are concatenated to form a positive binary





number which specifies how many bytes of data USER wants transferred. The seventh and eighth bytes are concatenated and used by USRUN to count the number of bytes that have been transferred. When the required number of characters have been transferred, or if USRUN terminates a READ or WRITE due to an abnormal condition, then USRUN calls a subroutine at an address defined by the ninth and tenth bytes of the command block. This subroutine, which is provided by USER, must determine the state of the process and then take appropriate action.

Since USRUN must be capable of operation in a full duplex mode (i.e., be able to receive and transmit simultaneously), it keeps the address of two control blocks; one for a READ operation and one for a WRITE. The address of the controlling command block is kept in RAM locations labeled RCBA for the READ operation and TCBA for the WRITE operation. If RCBA (Receive Control Block Address) or TCBA (Transmit Control Block Address) is zero, it indicates that the corresponding operation is in an idle status.

Flowcharts of USRUN appear in Figure 13 and the listings appear in Figure 14. The first section of the flowcharts (Figures 13.1 and 13.2) consists of two subroutines which are used as convenient tools for operating on the control blocks. These routines are labeled LOADA and CLEAN. LOADA is entered with the address of a control block in registers H and L. Upon return registers D and E have been set equal to the address in the buffer which is the target of the next data transfer (i.e.,  $D,E = BAD + CCT$ ); and CCT (transferred byte count) has then been incremented. In addition, the B register is set to zero if the number of bytes that have been transferred is equal to the number requested (i.e.,  $CCT = RCT$ ). CLEAN, the second routine, is also entered with the address of a command block in the H and L registers. In addition, the Accumulator holds the status which will be placed in the STATUS byte of the command block. On exit the STATUS byte has been updated and the address of the completion routine has been placed in H and L.

Upon interrupt, control of the MCS-80 system is transferred to VECTOR (Figure 13.3). VECTOR is a program which saves the state of the system, gets the status of the USART and jumps to the RISR (Receive Interrupt Service Routine) or the TISR (Transmit Interrupt Service Routine), depending on which of the two ready flags is active. If neither ready flag is active, VECTOR restores the status of the running program, enables interrupts, and returns. (Interrupts are automatically disabled by the hardware upon an interrupt.) This exit from VECTOR, which is labeled VOUT, is used from other

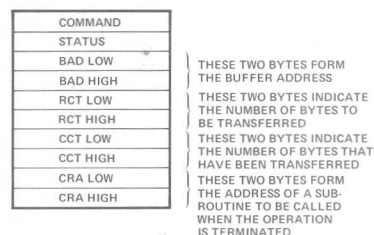


Figure 12. Control Block

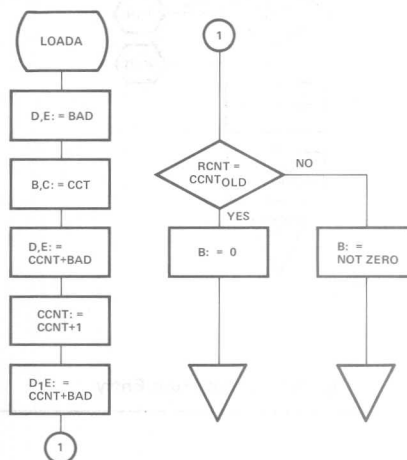


Figure 13.1. LOADA Subroutine

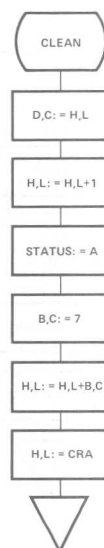


Figure 13.2. CLEAN Subroutine

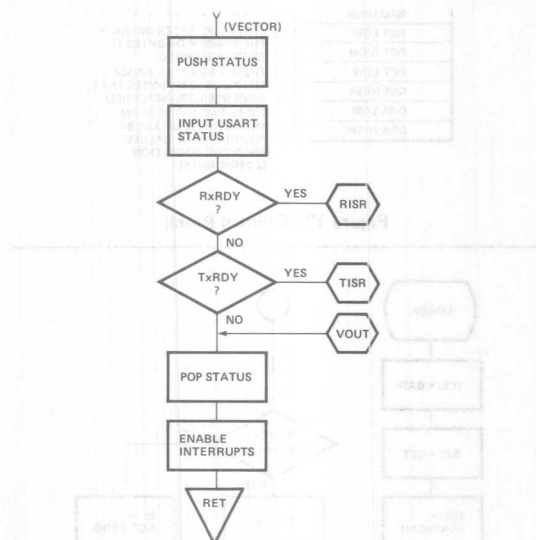


Figure 13.3. Interrupt Entry

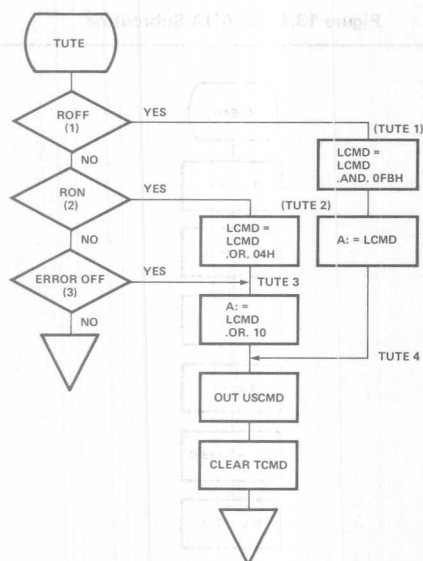


Figure 13.4. Transmit Interrupt Service Routine

In addition to handling normal data transfers, TISR (Figure 13.4) checks a location in memory named TCMD in order to determine if the receive program wishes to send a command to the USART. Since the transmit data and command must share a buffer within the USART, any command output must occur when TxRDY is asserted. If TCMD is zero, TISR proceeds with the data transfer. If TCMD is non-zero, TISR calls TUTE (Transmit Utility, Figure 13.5) which, depending on the value

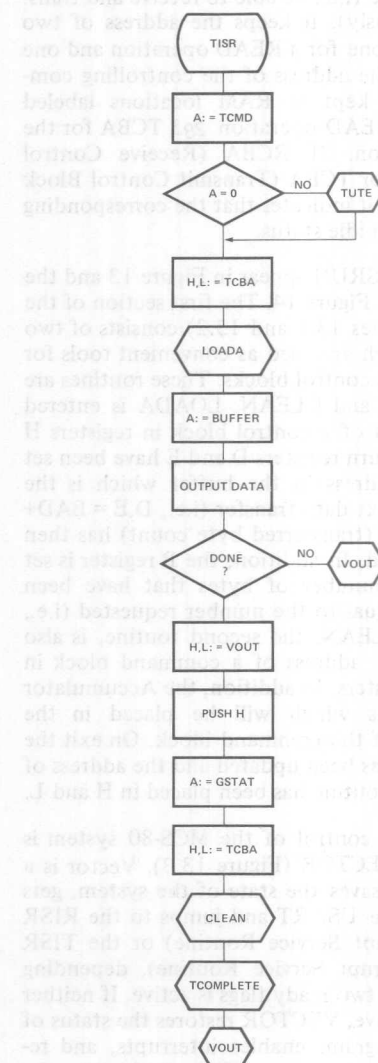


Figure 13.5. Transmit Utility Routine

The flowchart of the RISR is shown in Figure 13.6. Note that in addition to terminating whenever the required number of characters have been received, the RISR also terminates if one of the error flags becomes set or if the received character matches a character found in a table pointed to by the label ETAB. This table, which starts at ETAB and continues until an all "ones" entry is found, can be used by USER to define special characters, such as EOT (End Of Transmission), which will terminate a READ operation. The remainder of Figure 13 (13.7) shows the decoding of the commands to USRUN. The listings also include a test USER which exercises USRUN. This program sets up a 256-byte transmit buffer and transfers it to a similar input buffer by means of a local loop. When both the READ and WRITE operations are complete, the test USER checks to insure that the two buffers are identical. If the buffers differ, the MDS monitor is called; if the data is correct, the test is repeated.

The 8251 USART has been described both as a device and as a component in a system. Since not only modems but also many peripheral devices have a serial interface, the 8251 is an extremely useful component in a microcomputer system. A particular advantage of the device is that it is capable of operating in various modes without requiring hardware modifications to the system of which it is a part. As with any complex subsystem, however, the 8251 USART must be carefully applied so that it can be utilized to full advantage in the overall system. It is hoped that this application note will aid in the designer in the application of the 8251 USART. As a further aid to the application of the 8251, the appendix of this document includes a list of design hints based on past experience with the 8251.



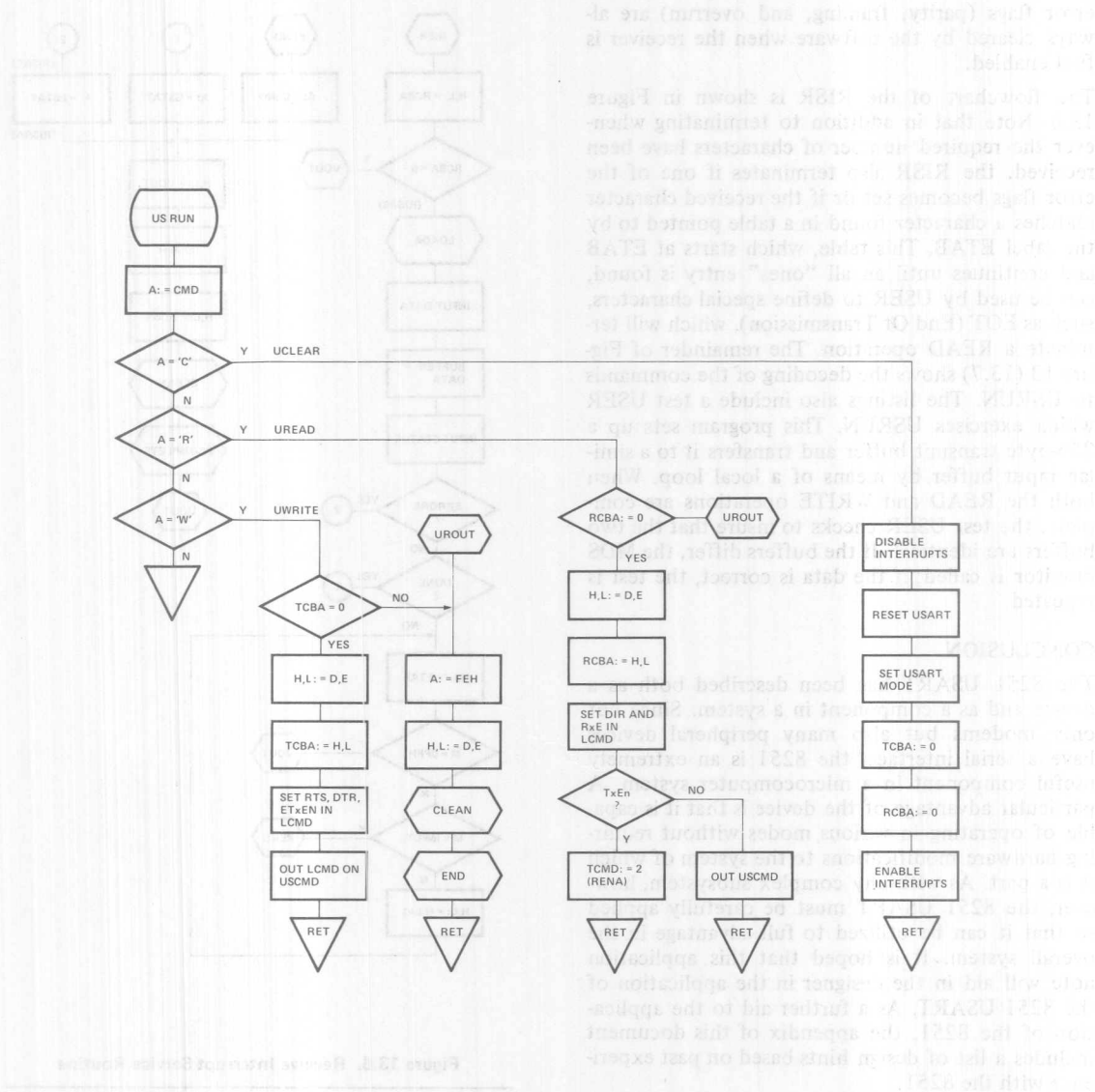


Figure 13.7. URUN Command Decode

Figure 14. Program Listing

```

;*****
;
;      SYSTEM ORIGIN STATEMENT
;
;*****
4000      ORG 4000H

;*****
;
;      DATA STORAGE FOR TEST USER
;
;*****

4000      BUFIN:  DS      100H      ;INPUT BUFFER
4100      BUFOUT: DS      100H      ;OUTPUT BUFFER
4200 5200  RBLOCK: DB      'R',00H  ;RECEIVE CONTROL BLOCK
4202 0040  RBAD:   DW      BUFIN
4204 FF00  RRCT:   DW      OFFH
4206 0000  RCCT:   DW      00H
4208 1742  RCRA:   DW      RCR
420A 5700  TBLOCK: DB      'W',00H ;TRANSMIT CONTROL BLOCK
420C 0041  TBAD:   DW      BUFOUT
420E FF00  TRCT:   DW      OFFH
4210 0000  TCCT:   DW      00H
4212 2742  TCRA:   DW      TCR
4214 4300  GBLOCK: DB      'C',00H
4216 00     FLAG:   DB      00H

;*****
;
;      COMPLETION ROUTINES
;
;*****

4217 AF      RCR:   XRA      A      ;CLEAR A
4218 323B42   STA      RCBA      ;TURN OFF RECEIVE
421B 323C42   STA      RCBA+1
421E 3A1642   LDA      FLAG      ;GET FLAG
4221 E60F     ANI      OFH       ;CLEAR UPPER FOUR BITS
4223 321642   STA      FLAG      ;RESTORE FLAG
4226 C9      RET
4227 AF      TCR:   XRA      A      ;CLEAR A
4228 323942   STA      TCBA      ;TURN OFF TRANSMIT
422B 323A42   STA      TCBA+1
422E 3A1642   LDA      FLAG      ;GET FLAG
4231 E6F0     ANI      OF0H      ;CLEAR LOWER FOUR BITS
4233 321642   STA      FLAG      ;RESTORE FLAG
4236 C9      RET      ;THEN RETURN

```

```

;
;*****
00F5      USTAT  EQU    0F5H      ;USART STATUS ADDRESS
00F5      USCMD  EQU    0F5H      ;USART CMD ADDRESS
00F4      USDAI  EQU    0F4H      ;USART DATA INPUT ADDRESS
00F4      USDAO  EQU    0F4H      ;USART DATA OUTPUT ADDRESS
0000      GSTAT  EQU    00H       ;GOOD STATUS
00FF      BSTAT  EQU    0FFH      ;BAD STATUS
0001      CEND   EQU    01H

```

```

;*****
;
;
;*****

```

# SYSTEM DATA TABLE

```

4237 00      LCMD: DB    00H      ;CURRENT OPERATING COMMAND
4238 00      TCMD: DB    00H      ;IF NON ZERO A COMMAND TO BE SENT
4239 0000     TCBA: DW    00H      ;ADDRESS OF XMIT CBLOCK
423B 0000     RCBA: DW    00H      ;ADDRESS OF RECEIVE CBLOCK
423D FF      MTAB: DB    0FFH      ;END CHARACTER TABLE

```

```

;*****
;
;
;
;
;
;
;
;
;
;*****

```

```

LOAD ADDRESS ROUTINE
LOADA IS ENTERED WITH THE ADDRESS OF A CONTROL
BLOCK IN H,L. ON EXIT D,E CONTAINS THE ADDRESS
WHICH IS THE TARGET OF THE NEXT DATA TRANSFER (BAD+CCNT)
AND B HAS BEEN SET TO ZERO IF THE REQUESTED NUMBER OF
TRANSFERS HAS BEEN ACCOMPLISHED. CCNT IS INCREMENTED
AFTER THE TARGET ADDRESS HAS BEEN CALCULATED.

```

```

423E 23      LOADA: INX      H          ;D,E GETS BUFFER ADDRESS
423F 23      INX      H          ;D,E GETS BUFFER ADDRESS
4240 5E      MOV      E,M
4241 23      INX      H          ;POINT AT TARGET
4242 56      MOV      D,M        ;DONE
4243 23      INX      H          ;B,C GETS COMPLETED COUNT (CCNT)
4244 23      INX      H          ;B,C GETS COMPLETED COUNT (CCNT)
4245 23      INX      H          ;B,C GETS COMPLETED COUNT (CCNT)
4246 4E      MOV      C,M        ;POINT AT TARGET
4247 23      INX      H          ;B,C GETS COMPLETED COUNT (CCNT)
4248 46      MOV      B,M        ;DONE
4249 EB      XCHG          ;D,E GETS BAD+CCNT
424A 09      DAD      B
424B EB      XCHG          ;DONE
424C 03      INX      B          ;CCNT GETS INCREMENTED
424D 70      MOV      M,B
424E 2B      DCX      H
424F 71      MOV      M,C        ;DONE
4250 0B      DCX      B          ;DOES OLD CCNT=RCNT?
4251 2B      DCX      H
4252 7E      MOV      A,M        ;AFTER THEN DO A
4253 90      SUB      B          ;BETTER WITHHOLDING
4254 47      MOV      B,A        ;BETTER WITHHOLDING
4255 C0      RNZ          ;NO-RETURN WITH B NOT ZERO
4256 2B      DCX      H
4257 7E      MOV      A,M
4258 91      SUB      C
4259 47      MOV      B,A
425A C9      RET          ;RETURN WITH B=0 IF RCNT=CCNT

```

```

; *****
;
; CLEAN-UP ROUTINE
; CLEAN IS ENTERED WITH THE ADDRESS OF A CONTROL
; BLOCK IN H,L AND A NEW STATUS TO BE
; ENTERED INTO IT IN A. ON EXIT THE ADDRESS OF THE
; CONTROL BLOCK IS IN D,E; THE STATUS OF THE BLOCK
; HAS BEEN UPDATED; AND THE ADDRESS OF THE COMPLETION
; ROUTINE IS IN H,L.
; *****

425B 5D      CLEAN: MOV     E,L      ;SAVE THE ADRESS OF THE COMMAND BLOCK
425C 54      MOV     D,H
425D 23      INX     H      ;POINT AT STATUS
425E 77      MOV     M,A      ;SET STATUS EQUAL TO A
425F 010700  LXI     B,7      ;SET INDEX TO SEVEN
4262 09      DAD     B      ;POINT AT COMPLETION ADDRESS
4263 7E      MOV     A,M      ;GET LOWER ADDRESS
4264 23      INX     H      ;POINT AT UPPER ADDRESS
4265 66      MOV     H,M      ;H GETS HIGH ADDRESS BYTE
4266 6F      MOV     L,A      ;L GETS LOW ADDRESS BYTE
4267 C9      RET

; *****
;
; INTERRUPT VECTOR ROUTINE
; VECTOR SAVES THE STATUS OF THE RUNNING PROGRAM
; THEN READS THE STATUS OF THE USART TO DETERMINE
; IF A RECEIVE OR TRANSMIT INTERRUPT OCCURRED.
; VECTOR THEN CALLS THE APPROPRIATE SERVICE ROUTINE.
; IF NEITHER INTERRUPTS OCCURRED THEN VECTOR RESTORES
; THE STATUS OF THE RUNNING PROGRAM. THE SERVICE
; ROUTINES USE THE EXIT CODE, LABLED VOUT, TO EFFECT
; THEIR EXIT FROM INTERRUPT MODE.
; *****

4268 F5      VECTOR: PUSH    PSW      ;PUSH STATUS INTO THE STACK
4269 C5      PUSH    B
426A D5      PUSH    D
426B E5      PUSH    H
426C DBF5    IN      USTAT      ;GET USART ADDRESS
426E DBFA    IN      OFAH      ;MDS-GET MONITOR CARD INT. STATUS
4270 0F      RRC          ;ROTATE TWO PLACES
4271 0F      RRC          ;SO THAT CARRY=RXRDY
4272 DA8842  JC      RISR      ;IF RXRDY GO TO SERVICE ROUTINE
4275 07      RLC          ;IF NOT ROTATE BACK
4276 07      RLC          ;LEAVING TXRDY IN CARRY
4277 DAD442  JC      TISR      ;IF TXRDY THEN GO TO SERVICE ROUTINE
427A 3EFC    MVI     A,OFCH    ;MDS-CLEAR OTHER LEVEL THREE INTERRUPTS
427C D3F3    OUT     OF3H      ;MDS
427E E1      VOUT: POP     H      ;ELSE EXIT FROM INTERRUPT MODE
427F D1      POP     D
4280 C1      POP     B
4281 3E20    MVI     A,20H      ;MDS-RESTORE CURRENT LEVEL
4283 D3FD    OUT     OFDH      ;MDS
4286 FB      EI          ;ENABLE INTERRUPTS
4287 C9      RET

```

```

;*****
;
; RECEIVE INTERRUPT SERVICE ROUTINE;
; RISR PROCESSES A RECEIVE INTERRUPT
; AT THE END OF RECEIVE THE USER SUPPLIED
; COMPLETION ROUTINE IS CALLED AND THEN AN
; EXIT IS TAKEN THROUGH VOUT OF THE
; VECTOR
;*****

4288 2A3B42 RISR:  LHL  RCBA
428B 3E82      MVI  A,82H ;MDS-CLEAR RECEIVE INTERRUPT
428D D3F3      OUT  OF3H ;MDS
428F 2C        INR  L
4290 2D        DCR  L
4291 C29942    JNZ  RISRB
4294 24        INR  H
4295 25        DCR  H
4296 CA7E42    JZ   VOUT
4299 CD3E42    RISRB: CALL  LOADA ;READY-SET UP ADDRESS
429C DBF4      IN   USDAI ;GET INPUT DATA
429E 12        STAX D ;AND PUT IN THE BUFFER
429F 4F        MOV  C,A ;SAVE INPUT DATA IN C
42A0 DBF5      IN   USTAT ;GET STATUS AGAIN
42A2 E638      ANI  38H ;MASK FOR ERROR FIELD
42A4 C2B942    JNZ  RISRE ;NOT ZERO-TAKE ERROR EXIT
42A7 04        INR  B ;B WAS 00 IF DONE
42A8 05        DCR  B
42A9 C2BE42    JNZ  EXCHAR ;NOT DONE-EXIT
42AC 3E00      MVI  A,GSTAT ;A GETS GOOD STATUS
42AE 217E42    RISRA: LXI  H,VOUT ;GET RETURN ADDRESS
42B1 E5        PUSH H ;AND PUSH IT INTO THE STACK
42B2 2A3B42    LHL  RCBA ;POINT H,L AT THE CMD BLOCK
42B5 CD5B42    CALL CLEAN ;CALL CLEANUP ROUTINE
42B8 E9        PCHL ;EFFECTIVELY CALLS COMPLETION ROUTINE
;RETURN IS TO VOUT BECAUSE OF PUSH H

42B9 3EFF      RISRE: MVI  A,BSTAT ;A GETS BAD STATUS
42BB C3AE42    JMP  RISRA ;OTHERWISE EXIT IS NORMAL
42BE 213D42    EXCHAR: LXI  H,MTAB ;TEST CHARACTER AGAINST EXIT TABLE
42C1 7E        EXA:  MOV  A,M
42C2 FEFF      CPI  OFFH ;END OF TABLE
42C4 CA7E42    JZ   VOUT
42C7 B9        CMP  C
42C8 CACF42    JZ   PEND ;MATCH-TERMINATE READ
42CB 23        INX  H
42CC C3C142    JMP  EXA
42CF 3E01      PEND:  MVI  A,CEND
42D1 C3AE42    JMP  RISRA

```

```

;*****
;
; TRANSMIT INTERRUPT SERVICE ROUTINE
; TISR PROCESSES TRANSMITTER INTERRUPTS
; WHEN THE END OF A TRANSMISSION IS
; DETECTED THE USER SUPPLIED COMPLETION
; ROUTINE IS CALLED AND THEN AN EXIT IS
; TAKEN THROUGH VOUT OF VECTOR
;*****

42D4 3A3842 TISR: LDA TCMD ;GET POTENTIAL COMMAND
42D7 B7 ORA A ;DESIGNATE ON IT
42D8 C40443 CNZ TUTE ;DO UTILITY COMMAND
42DB 3E81 MVI A,081H ;MDS-CLEAR XMIT INTERRUPTS
42DD D3F3 OUT OF3H ;MDS
42DF 2A3942 LHLD TCBA
42E2 2C INR L ;MAKE SURE HAVE VALID CONTROL BLOCK
42E3 2D DCR L
42E4 C2EC42 JNZ TISRA ;GOOD
42E7 24 INR H
42E8 25 DCR H
42E9 CA7E42 JZ VOUT ;NON VALID BLOCK (H,L=0)
42EC CD3E42 TISRA: CALL LOADA ;SET UP ADDRESS
42EF 1A LDAX D ;GET DATA FROM BUFFER
42F0 D3F4 OUT USDAO ;AND OUTPUT IT
42F2 04 INR B ;B WAS 00 IF DONE
42F3 05 DCR B
42F4 C27E42 JNZ VOUT ;NOT DONE-EXIT FROM SERVICE ROUTINE
42F7 217E42 LXI H,VOUT ;SET UP RETURN ADDRESS
42FA E5 PUSH H ;AND PUSH IT INTO THE STACK
42FB 3E00 MVI A,GSTAT ;A GETS GOOD STATUS
42FD 2A3942 LHLD TCBA ;POINT H,L AT COMMAND BLOCK
4300 CD5B42 CALL CLEAN ;CALL CLEANUP ROUTINE
4303 E9 PCHL ;CALL COMPLETION ROUTINE
;RETURN WILL BE TO VOUT

4304 FE01 TUTE: CPI 01 ;RECEIVER OFF
4306 CA2443 JZ TUTE1
4309 FE02 CPI 02 ;RECEIVER ON
430B CA1443 JZ TUTE2
430E FE03 CPI 03 ;CLEAR ERRORS
4310 CA1C43 JZ TUTE3
4313 C9 RET
4314 3A3742 TUTE2: LDA LCMD
4317 F604 ORI 04
4319 323742 STA LCMD
431C 3A3742 TUTE3: LDA LCMD
431F F610 ORI 10H
4321 D3F5 TUTE4: OUT USCMD
4323 C9 RET
4324 3A3742 TUTE1: LDA LCMD
4327 E6FB ANI OFBH
4329 323742 STA LCMD
432C C32143 JMP TUTE4

```

```

;*****
;
;      USART COMMAND BLOCK INTERPRETER
;      USRUN IS CALLED BY USER WITH THE ADDRESS
;      OF THE COMMAND BLOCK IN H,L. USRUN EXAMINES
;      THE BLOCK AND INITIALIZES THE REQUESTED OPERATION
;*****

432F 1A      USRUN: LDAX D      ;GET THE CMD FROM THE BLOCK
4330 FE43    CPI 'C'      ;IS IT A CLEAR COMMAND?
4332 CA4043  JZ UCLEAR    ;YES GO TO CLEAR ROUTINE
4335 FE52    CPI 'R'      ;IS IT A READ COMMAND?
4337 CA5D43  JZ UREAD     ;YES-GO TO READ ROUTINE
433A FE57    CPI 'W'      ;IS IT A WRITE COMMAND?
433C CA9D43  JZ UWRITE    ;GO TO WRITE ROUTINE
433F C9      RET          ;NOT A GOOD COMMAND-RETURN
4340 F3      UCLEAR: DI    ;DISABLE INTERRUPTS
4341 AF      XRA A        ;CLEAR A
4342 D3F5    OUT USCMD    ;OUTPUT THREE TIMES TO ENSURE
4344 D3F5    OUT USCMD    ;THAT THE USART IS IN A KNOWN STATE
4346 D3F5    OUT USCMD
4348 3E40    MVI A,40H    ;CODE TO RESET USART
434A D3F5    OUT USCMD    ;OUTPUT ON CMD CHANNEL
434C 3E5E    MVI A,05EH   ;CE IMPLIES ASYN MODE (X16)
                        ;      8 DATA BITS
                        ;      ODD PARITY
                        ;      1 STOP BIT

434E D3F5    OUT USCMD    ;OUTPUT ON CMD CHANNEL
4350 AF      XRA A        ;CLEAR A, SET ZERO
4351 213942  LXI H,TCBA   ;CLEAR TCBA AND RCBA
4354 77      MOV M,A
4355 23      INX H
4356 77      MOV M,A
4357 23      INX H
4358 77      MOV M,A
4359 23      INX H
435A 77      MOV M,A
435B FB      EI          ;ENABLE INTERRUPTS
435C C9      RET          ;AND RETURN TO USER
                        ;
                        ;

435D 213B42  UREAD: LXI H,RCBA ;CHECK READ IDLE
4360 7E      MOV A,M
4361 B7      ORA A
4362 C26B43  JNZ UROUT
4365 23      INX H
4366 7E      MOV A,M
4367 B7      ORA A
4368 CA7743  JZ URDA      ;READ IS IDLE-PROCEED
436B 3EFE    UROUT: MVI A,0FEH ;ALREADY RUNNING-ERROR STATUS
436D 217643  LXI H,URDB   ;SET UP RETURN ADDRESS
4370 E5      PUSH H       ;PUSH IT INTO STACK
4371 EB      XCHG          ;H GETS COMMAND BLOCK ADDRESS
4372 CD5B42  CALL CLEAN    ;CALL CLEANUP ROUTINE
4375 E9      PCHL          ;EFFECTIVELY CALLS END ROUTINE
4376 C9      URDB: RET     ;RETURN TO USER

4377 EB      URDA: XCHG     ;H GETS COMMAND BLOCK ADDRESS
4378 223B42  SHLD RCBA     ;RCBA GETS COMMAND BLOCK ADDRESS
437B 3A3742  LDA LCMD      ;GET LAST COMMAND
437E F616    ORI 16H      ;SET RXE AND DTR AND RESET ERRORS
4380 323742  STA LCMD     ;AND RETURN TO MEMORY
4383 0F      RRC          ;SET CARRY EQUAL TO TXE

```

4387	3E02	MVI	A,2	
4389	323842	STA	TCMD	
438C	07	URDC:	RLC	
438D	D3F5	OUT	USCMD	;OUTPUT CMD
438F	DBF4	IN	USDAI	;CLEAR USART OF LEFT OVER CHARACTERS
4391	DBF4	IN	USDAI	
4393	3E82	MVI	A,82H	;MDS-CLEAR RECEIVE INTERRUPT
4395	D3F3	OUT	OF3H	;MDS
4397	3EF6	MVI	A,0F6H	;MDS-ENABLE LEVEL THREE
4399	D3FC	OUT	OFCH	;MDS
439B	FB	EI		;ENABLE INTERRUPTS
439C	C9	RET		;RETURN TO USER
439D	213942	UWRITE:	LXI	H,TCBA ;CHECK WRITE IDLE
43A0	7E	MOV	A,M	
43A1	B7	ORA	A	
43A2	C26B43	JNZ	UROUT	;BUSY-EXIT
43A5	23	INX	H	
43A6	7E	MOV	A,M	
43A7	C26B43	JNZ	UROUT	;BUSY-EXIT
43AA	EB	XCHG		;OK-H GETS COMMAND BLOCK ADDRESS
43AB	223942	SHLD	TCBA	;TCBA GETS COMMAND BLOCK ADDRESS
43AE	3A3742	LDA	LCMD	;GET LAST COMMAND
43B1	F623	ORI	023H	;SET RTS,DTR, AND TXEN
43B3	323742	STA	LCMD	
43B6	D3F5	OUT	USCMD	
43B8	3EF6	MVI	A,0F6H	;MDS-ENABLE LEVEL THREE INTERRUPTS
43BA	D3FC	OUT	OFCH	;MDS
43BC	FB	EI		;ENABLE SYSTEM INTERRUPTS
43BD	C9	RET		;AND RETURN

```

;*****
;
;      USER IS A TEST PROGRAM WHICH EXERCISES USRUN
;
;*****

43BE 3EC3      USER:  MVI      A,0C3H  ;MDS-SET INTERRUPT VECTOR
43C0 321800    STA      018H
43C3 216842    LXI      H,VECTOR
43C6 221900    SHLD     019H
43C9 3E43      MVI      A,'C'      ;SET GENERAL BLOCK TO A 'C'
43CB 111442    LXI      D,GBLOCK
43CE 12        STAX     D
43CF CD2F43    CALL     USRUN
43D2 210040    LXI      H,BUFIN  ;CLEAR INPUT BUFFER
43D5 AF        XRA      A
43D6 77        MOV      M,A
43D7 2C        INR      L
43D8 C2D643    JNZ      $-2
43DB 210041    LXI      H,BUFOUT ;INITIALIZE OUTPUT BUFFER
43DE 75        MOV      M,L
43DF 2C        INR      L
43E0 C2DE43    JNZ      $-2
43E3 65        MOV      H,L      ;REINTIALIZE CONTROL BLOCKS
43E4 2E52      MVI      L,'R'
43E6 220042    SHLD     RBLOCK
43E9 2E57      MVI      L,'W'
43EB 220A42    SHLD     TBLOCK
43EE 6C        MOV      L,H
43EF 220642    SHLD     RCCT
43F2 221042    SHLD     TCCT
43F5 110042    LXI      D,RBLOCK ;START READ
43F8 CD2F43    CALL     USRUN
43FB 110A42    LXI      D,TBLOCK ;START WRITE
43FE CD2F43    CALL     USRUN
4401 3EFF      MVI      A,OFFH  ;LOOP WAITING COMPLETION
4403 321642    STA      FLAG    ;FLAG WILL BE SET BY COMPLETION ROUTINES
4406 3A1642    LDA      FLAG
4409 B7        ORA      A
440A C20644    JNZ      $-4
440D 210040    LXI      H,BUFIN ;TEST INPUT BUFFER=OUTPUT BUFFER
4410 7E        COMLP:  MOV     A,M
4411 24        INR      H
4412 BE        CMP      M
4413 C21E44    JNZ      COMER
4416 25        DCR      H
4417 2C        INR      L
4418 C21044    JNZ      COMLP
441B C3BE43    JMP      USER    ;GOOD COMPARE-REPEAT TEST
441E C7        COMER:  RST      0      ;ERROR-RETURN TO MONITOR

0000      END

```

BSTAT 00FF	BUFIN 4000	BUFOU 4100	CEND 0001
CLEAN 425B	COMER 441E	COMLP 4410	EXA 42C1
EXCHA 42BE	FLAG 4216	GBLOC 4214	GSTAT 0000
LCMD 4237	LOADA 423E	MTAB 423D	PEND 42CF
RBAD 4202	RBLOC 4200	RCBA 423B	RCCT 4206
RCR 4217	RCRA 4208	RISR 4288	RISRA 42AE
RISRB 4299	RISRE 42B9	RRCT 4204	TBAD 420C
TBLOC 420A	TCBA 4239	TCCT 4210	TCMD 4238
TCR 4227	TCRA 4212	TISR 42D4	TISRA 42EC
TRCT 420E	TUTE 4304	TUTE1 4324	TUTE2 4314
TUTE3 431C	TUTE4 4321	UCLEA 4340	URDA 4377
URDB 4376	URDC 438C	UREAD 435D	UROUT 436B
USCMD 00F5	USDAI 00F4	USDAO 00F4	USER 43BE
USRUN 432F	USTAT 00F5	UWRIT 439D	VECTO 4268
VOUT 427E			

## APPENDIX A

### 8251 DESIGN HINTS

1. Output of a command to the USART destroys the integrity of a transmission in progress if timed incorrectly.

Sending a command into the USART will overwrite any character which is stored in the buffer waiting for transfer to the parallel-to-serial converter in the device. This can be avoided by waiting for TxRDY to be asserted before sending a command if transmission is taking place. Due to the internal structure of the USART, it is also possible to disturb the transmission if a command is sent while a SYN character is being generated by the device. (The USART generates a SYN if the software fails to respond to TxRDY.) If this occurrence is possible in a system, commands should be transferred only when a positive-going edge is detected on the TxRDY line.

2. RxE only acts as a mask to RxRDY; it does not control the operation of the receiver.

When the receiver is enabled, it is possible for it to already contain one or two characters. These characters should be read and discarded when the RxE bit is first set. Because of these extraneous characters the proper sequence for gaining synchronization is as follows:

1. Disable interrupts
2. Issue a command to enter hunt mode, clear errors, and enable the receiver (EH,ER,RxE=1)
3. Read USART data (it is not necessary to check status)
4. Enable interrupts

The first RxRDY that occurs after the above sequence will indicate that the SYN character or

characters have been detected and the next character has been assembled and is ready to be read.

3. Loss of CTS or dropping TxEnable will immediately clamp the serial output line.

TxEnable and RTS should remain asserted until the transmission is complete. Note that this implies that not only has the USART completed the transfer of all bits of the last character, but also that they have cleared the modem. A delay of 1 msec following a proper occurrence of TxEmpty is usually sufficient (see item 4). An additional problem can occur in the synchronous mode because the loss of TxEnable clamps the data in at a SPACE instead of the normal MARK. This problem, which does not occur in the asynchronous mode, can be corrected by an external gate combining RTS and the serial output data.

4. Extraneous transitions can occur on TxEmpty while data (including USART generated SYNs) is transferred to the parallel-to-serial converter.

This situation can be avoided by ensuring that TxEmpty occurs during several consecutive status reads before assuming that the transmitter is truly in the empty state.

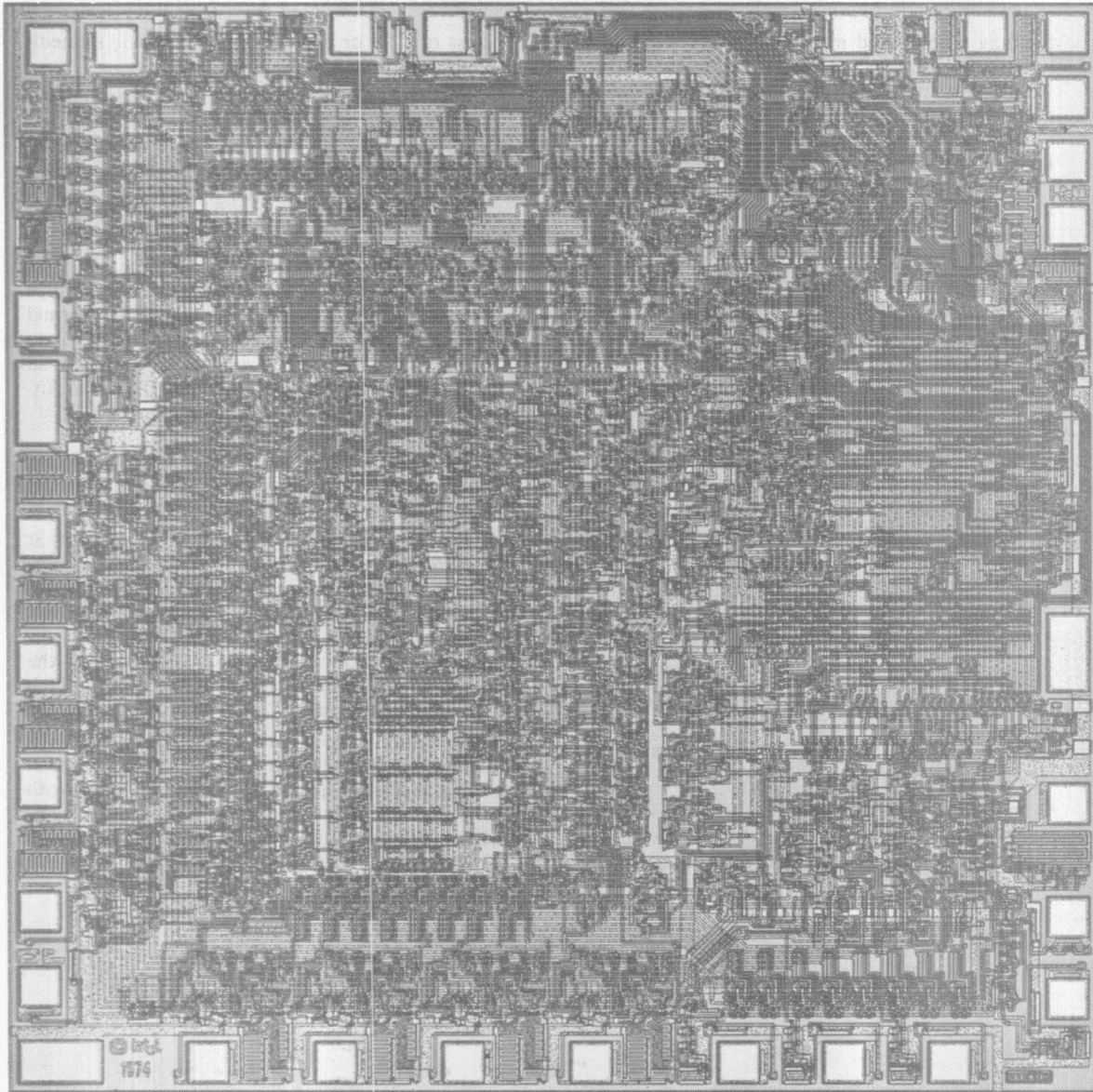
5. A BREAK (i.e., long space) detected by the receiver results in a string of characters which have framing errors.

If reception is to be continued after a BREAK, care must be taken to ensure that valid data is being received; special care must be taken with the last character perceived during a BREAK, since its value, including any framing error associated with it, is indeterminate.

# 8251 PROGRAMMABLE COMMUNICATION INTERFACE

## APPENDIX A 8251 DESIGN HINTS

The output of a command to the USART device is the integrity of a transmission in progress. If characters have been detected and the next character has been assembled and is ready to be read.

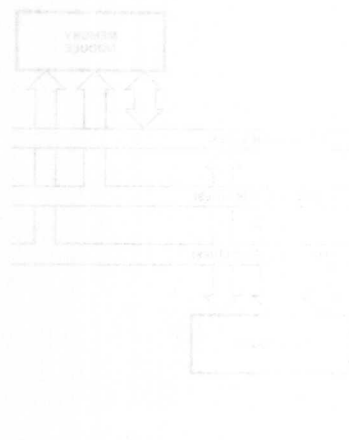


# 8255A Programmable Peripheral Interface Applications

by Alan Ebright

INTRODUCTION.....	2-64
OVERVIEW.....	2-64
8080 CPU MODULE INTERFACE.....	2-64
PERIPHERAL INTERFACE SECTION.....	2-66
INTERNAL LOGIC SECTION.....	2-67
Mode Definition.....	2-67
Bit Set/Reset.....	2-68
INTERRUPT CONTROL LOGIC STATUS WORDS.....	2-69
SOFTWARE CONSIDERATION.....	2-71
MODE 0 — STATUS DRIVEN PERIPHERAL INTERFACE.....	2-73
8255A To Peripheral Hardware Interface.....	2-73
8080 CPU Module To 8255A Interface.....	2-73
Mode 0 Interface Software.....	2-75
Summary/Conclusions.....	2-76
MODE 1 — INTERRUPT DRIVEN PRINTER INTERFACE.....	2-78
CPU Module To 8255A Interface.....	2-78
8255A To Peripheral Interface.....	2-78
Mode 1 Software Driver.....	2-79
Summary/Conclusions.....	2-80
MODE 2 — 8080 TO 8080 INTERFACE.....	2-83
Hardware Discussion.....	2-83
Software Discussion.....	2-83
Summary/Conclusions.....	2-83
APPENDIX A — 8255A QUICK REFERENCE.....	2-90

The 8255A is a complete peripheral interface module for the 8080 CPU. It is designed to be connected to the 8080 CPU via a single 80-pin DIP package. The 8255A contains three 8-bit parallel ports, each of which can be configured as either input or output. It also includes a programmable interrupt controller (PIC) and a status register. The 8255A is designed to be used in a variety of applications, including data acquisition, process control, and communications.



## INTRODUCTION

Microprocessor-based system designs are a cost-effective solution to a wide variety of problems. When a system designer is presented with the task of selecting a microprocessor for a design, the capabilities of the microprocessor should not be the only consideration. The microprocessor should be an element of a compatible family of devices. The MCS-80 component family is a group of compatible devices which have been designed to directly address and solve the problems of microprocessor-based system design. One member of the MCS-80 component family is Intel's 8255A programmable peripheral interface chip. This device replaces a significant percentage of the logic required to support a variety of byte oriented Input/Output interfaces. Through the use of the 8255A, the I/O interface design task is significantly simplified, the design flexibility is increased, and the number of components required is reduced.

This application note presents detailed design examples from both the hardware and software points of view. Since the 8255A is an extremely flexible device, it is impossible to list all of the applications and configurations of the device. A number of designs are presented which may be modified to fulfill specific user interface requirements.

Detailed design examples are discussed within the context of the 8080 system shown in Figure 1. The basic 8080 system is composed of the CPU module, memory module, and the I/O module. CPU module and memory module design are discussed

within other Intel publications. This application note deals exclusively with I/O module design.

It is assumed that the reader is familiar with the MCS-80 User's Manual and/or the MCS-85 User's Manual, particularly the 8255A device description.

## OVERVIEW OF THE 8255A

The 8255A block diagram shown in Figure 2 has been divided into three sections: 8080 CPU Module Interface, Peripheral Interface, and the Internal Logic.

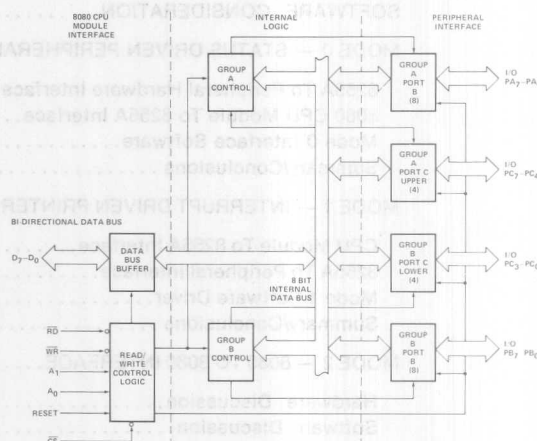
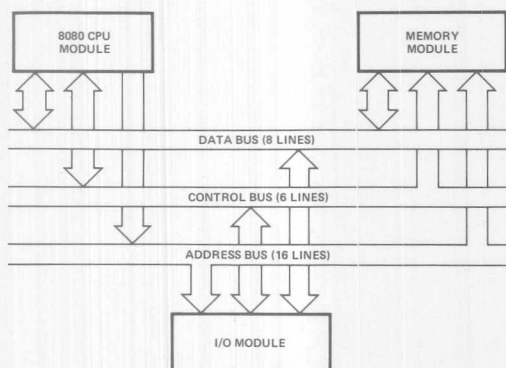


Figure 2. 8255A Block Diagram



## 8080 CPU MODULE INTERFACE

The 8255A is a compatible member of the MCS-80 component family and, therefore, may be directly interfaced to the 8080. Figure 3 displays one method of interconnecting the 8255A and an 8080 CPU module. The 8080 CPU module consists of the 8080A CPU, the 8224 Clock Generator, and the 8228 System Controller. The system shown in Figure 3 utilizes a linear select scheme which dedicates an address line as an exclusive enable (chip select) for each specific I/O device. The chip select signal is used to enable communication between the selected 8255A and the 8080 CPU. I/O Ports A, B, C, or the Control Word Register are selected by the two port select signals ( $A_1$ ,  $A_0$ ). These signals ( $A_1$  and  $A_0$ ) are driven by the least significant bits of the address bus. The I/O port select characters required by this configuration are shown in Figure 4.

When a system utilizing the linear select scheme is implemented, a maximum of six I/O devices may be selected. If more than six I/O devices must be addressed, the six device select bits must be encoded to generate a maximum of 64 device select lines. Note that when large systems are implemented, bus loading considerations may require that bus drivers be included in the CPU module. The MCS-80 component family contains parts which are designed to perform this function (8216, 8226).

The 8255A I/O read ( $\overline{RD}$ ) and I/O write ( $\overline{WR}$ ) signals may be directly driven by the 8228. This results in an isolated I/O architecture where 8080 Input/Output instructions are used to reference an independent I/O address space. An alternate approach is memory mapped I/O. This architecture treats an area of memory as the I/O address space. The memory mapped I/O architecture utilizes 8080 memory reference instructions to access the I/O address space. Interfacing with the 8080 is outlined in Chapter 3 of the "8080 Microcomputer User's Manual".

The most important feature of the 8255A to 8080 CPU Module Interface is that for small system designs the 8255A may be interfaced directly to

the standard MCS-80 component family with no external logic. Minimum external logic is required in large system designs.

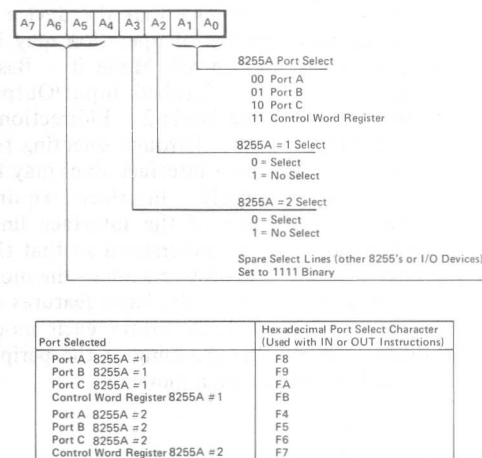


Figure 4. I/O Port Select Characters

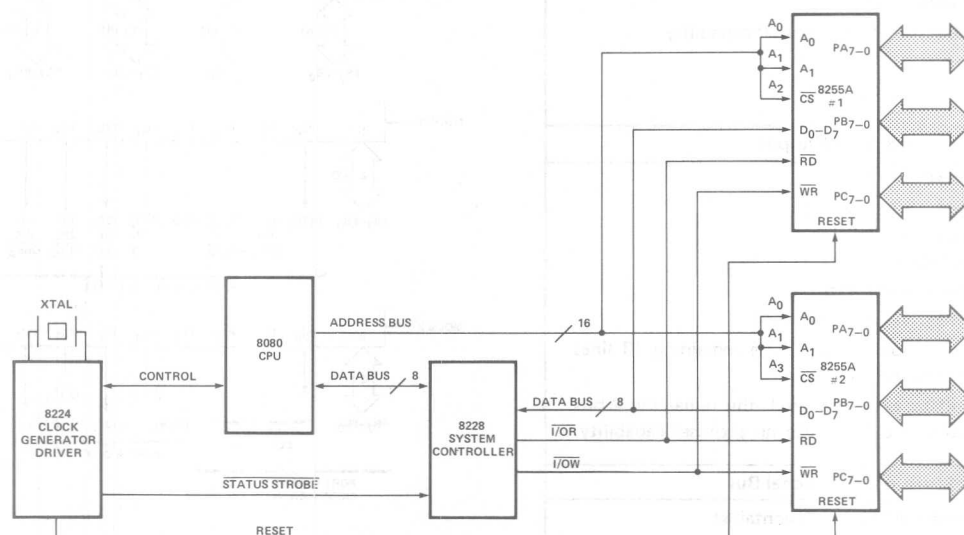


Figure 3. Linear Select 8255A Interconnect

## PERIPHERAL INTERFACE SECTION

The peripheral interface section contains 24 peripheral interface lines, buffers, and control logic. The characteristics and functions of the interface lines are determined by the operating mode selected under program control. The flexibility of the 8255A is due to the fact that the device is programmable. Three modes of operation may be selected under program control: Mode 0 — Basic Input/Output, Mode 1 — Strobed Input/Output with interrupt support, and Mode 2 — Bidirectional bus with interrupt support. Through selecting the correct operating mode, the interface lines may be configured to fulfill specific interface requirements. The characteristics of the interface lines within each mode must be understood so that the designer may utilize the 8255A to achieve the most efficient design. Table I lists the basic features of the peripheral interface lines within each mode group. Figure 5 shows the grouping of the peripheral interface lines within each mode.

Table I. Features of Peripheral Interface Lines

<b>Mode 0 — Basic Input/Output</b>
Two 8-bit ports
Two 4-bit ports with bit set/reset capability
Outputs are latched
Inputs are not latched
<b>Mode 1 — Strobed Input/Output</b>
One or two strobed ports
Each Mode 1 port contains:
8-bit data port
3 control lines
Interrupt support logic
Any port may be input or output
If one Mode 1 port is used, the remaining 13 lines may be configured in Mode 0.
If two Mode 1 ports are used, the remaining 2 bits may be input or output with bit set/reset capability.
<b>Mode 2 — Strobed Bidirectional Bus</b>
One bidirectional bus which contains:
8-bit bidirectional bus supported by Port A
5 control lines
Interrupt support logic
Inputs and outputs are latched
The remaining 11 lines may be configured in either Mode 0 or Mode 1.

One feature of Port C is important to note. Each Port C bit may be individually set and reset. Through the use of this feature, device strobes may be easily generated by software without utilizing external logic. The Mode 1 and Mode 2 configurations use a number of the Port C lines for interrupt control lines. Thus, the 8255A contains a large portion of the logic required to implement an interrupt driven I/O interface. This feature simplifies interrupt driven hardware design and saves a significant amount of the external logic that is normally required when less powerful I/O chips are used. In fact, the design examples contained in this application note describe how interrupt driven interfaces may be designed such that the only interrupt control logic required is that contained in the 8255A.

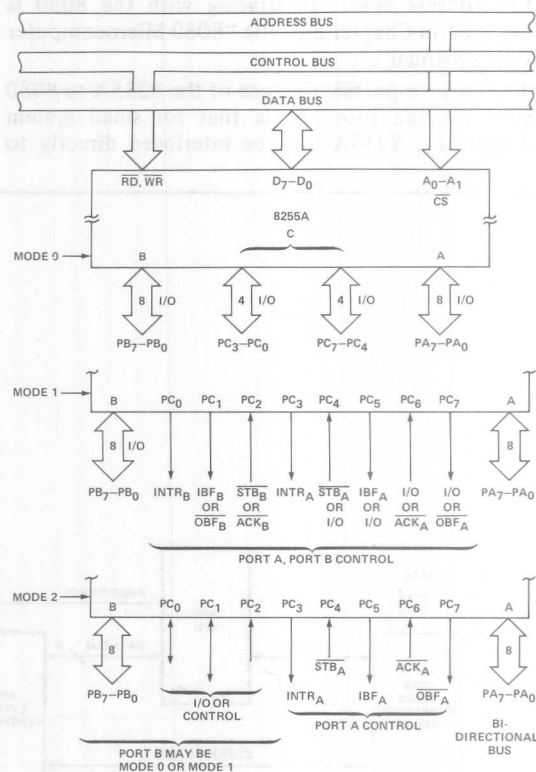


Figure 5. Grouping of Peripheral Interface Lines

## INTERNAL LOGIC SECTION

The internal logic section manages the transfer of data and control information on the internal data bus (refer to Figure 2). If the port select lines ( $A_1$  and  $A_0$ ) specify Ports A, B, or C, the operation is an I/O port data transfer. The internal logic will select the specified I/O port and perform the data transfer between the I/O port and the CPU interface. As was previously mentioned, both the functional configuration of each port and bit set/reset on Port C are controlled by the system's software. When the control word register is selected, the internal logic performs the operation described by the control word. The control word contains an opcode field which defines which of the two functions are to be performed (mode definition or bit set/reset).

### Mode Definition

When the opcode field (Bit 7) of the control word is equal to a one, the control word is interpreted by the 8255A as a mode definition control word. The mode definition control word (shown in Figure 6) is used to specify the configuration of the

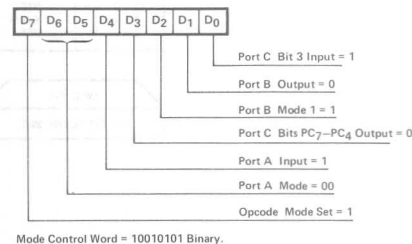
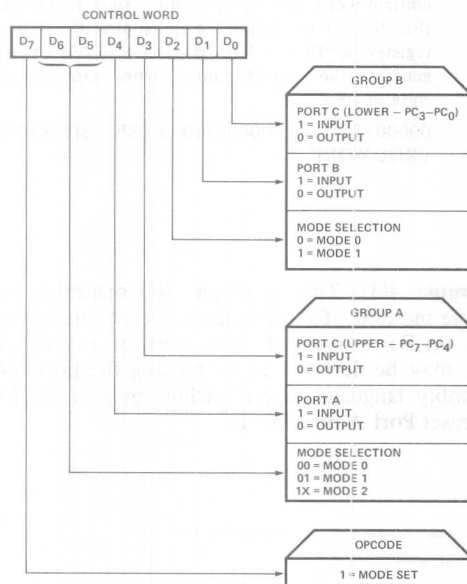
24 8255A peripheral interface lines. The system's software may specify the modes of Port A and Port B independently. Port C may be treated independently or divided into two portions as required by the Port A and Port B mode definitions.

**Example #1:** This example demonstrates how a mode control word is constructed and issued to an 8255A. The mode control word is passed to the device through the use of an output instruction that references an 8080 I/O port address. The value of the I/O port address is determined by the 8080 CPU interface implemented. This example references the I/O port addresses realized by the simple 8080 to 8255A interface shown in Figure 3.

If an 8255A is to be configured through the use of the mode control word interface as:

Port A	Mode 0 Input
Port B	Mode 1 Output
Port C	Bits $PC_7-PC_4$ Output
Port C	Bit 3 Input

The following mode control word is used:



The assembly language program is:

```

CWR      EQU    0FBH      ; 8255A #1 CONTROL WORD REGISTER
; *****
;      ISSUE MODE CONTROL WORD
; *****
MVI      A, 10010101B      ; GET MODE CONTROL WORD
OUT      CWR               ; OUTPUT TO 8255A #1 CONTROL WORD
                                ; REGISTER

```

Figure 6. Mode Definition Control Word

## Bit Set/Reset

When the opcode field (Bit 7) of the control word is equal to a zero, the control word is interpreted by the 8255A as a Port C bit set/reset command word (see Figure 7). Through the use of the bit set/reset command, any of the 8 bits on Port C may be independently set or reset. Note that control word bits 6–4 are not used. Bits 6–4 should be set to zero.

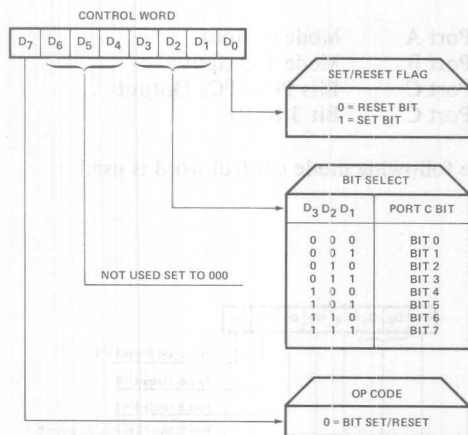
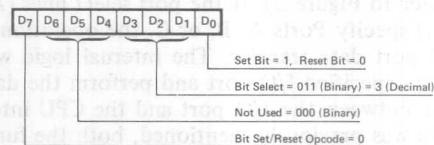


Figure 7. Bit Set/Reset Control Word

**Example #2:** This example demonstrates how a Port C bit set/reset control word is constructed and issued to an 8255A. The bit set/reset control word is passed to the device through the use of an output instruction that references an 8080 I/O port address. The value of the I/O port address is determined by the 8080 CPU interface implemented. This example references the I/O port addresses realized by the simple 8080 to 8255A interface shown in Figure 3.

Control word (see Figure 7).



The control word for set Port C bit 3 is 00000111 binary.  
The control word for reset Port C bit 3 is 00000110 binary.

The assembly language program is:

```
CWR EQU 0FBH ; 8255A #1 CONTROL WORD REGISTER
; SET BIT 3
MVI A, 00000111B ; GET SET BIT 3 CONTROL WORD
OUT CWR ; OUTPUT TO 8255A #1 CONTROL WORD REGISTER
; RESET BIT 3
MVI A, 00000110B ; GET RESET BIT 3 CONTROL WORD
OUT CWR ; OUTPUT TO 8255A #1 CONTROL WORD REGISTER
```

**NOTE:** An MVI instruction is used to load the reset bit 3 control word into the A register. Since it is known that the set bit control word is already in the A register, a "DCR A" instruction could be used to generate the correct control word and save one byte of code.

00000111 - 1 = 00000110 (RESET BIT 3 CONTROL WORD)

**Example #3:** This example demonstrates one simple method of performing a bit set/reset operation on Ports A and B. The state of any output port may be determined by reading the port. The assembly language program which may be used to set/reset Port A or B bits is:

```
PORTA EQU 0FBH ; 8255A #1 PORT A
; SET BIT 0
IN PORTA ; GET STATE OF PORT
ORI 01H ; SET BIT 0
OUT PORTA ; OUTPUT TO PORT
; RESET BIT 0
IN PORTA ; GET STATE OF PORT
ANI 0FEH ; RESET BIT 0
OUT PORTA ; OUTPUT TO PORT
```

## INTERRUPT CONTROL LOGIC STATUS WORDS

As previously mentioned, the 8255A Mode 1 and Mode 2 configurations support interrupt control logic. If a read of Port C is issued when the 8255A is configured in Mode 1, the software will receive the Mode 1 status word shown in Figure 8. The bits in the status word correspond to the state of the associated Port C lines (buffer full, interrupt request, etc.). The INTE bit shown in the status word corresponds to the interrupt enable flip-flop contained in the 8255A. This signal is not available externally. The structure of the Mode 1 status word varies as a function of the mode of the 8255A. Example #4 shows the status word which results from reading Port C from an 8255A which is configured with Port A Mode 1 input and Port B Mode 1 output.

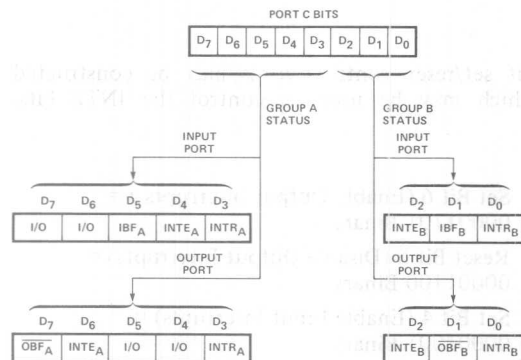


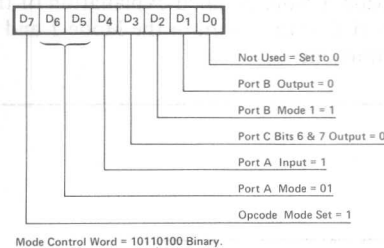
Figure 8. Mode 1 Status Word

### Example #4 — MODE 1 STATUS WORD

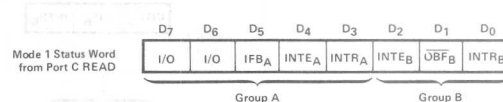
If an 8255A is to be configured through the use of the mode control word interface as:

Port A    Mode 1 Input  
Port B    Mode 1 Output  
Port C    Bits 6 & 7 Output

The following mode control word is used:



After the 8255A mode control word has been issued, a READ of Port C will obtain the following Mode 1 status word:



**NOTE:** The Port C I/O bits D7 and D6 should be modified through the use of the Port C bit set/reset command word. If a write to Port C is issued, the INTE<sub>A</sub> and INTE<sub>B</sub> bits may be inadvertently modified by the user. The IBF<sub>A</sub>, INTR<sub>A</sub>, OBF<sub>B</sub>, and INTR<sub>B</sub> bits will not be modified by either a write to Port C or a bit set/reset command. These four bits always reflect the state of the interrupt control logic.

Note that the Mode 2 status word (shown in Figure 9) differs from the Mode 1 status word. The format of the status word data bits D<sub>2</sub>–D<sub>0</sub> are defined by the specification of the Port B configuration. Example #5 shows the structure of the Mode 2 status word when the 8255A is configured with Port A Mode 2 (bidirectional bus) and Port B Mode 1 input.

The Mode 1 and Mode 2 status words reflect the state of the interrupt logic supported by the 8255A.

---

2011 年 11 月 11 日

INTE<sub>2</sub> – Bit 4 – Enable input interrupts

© 2004 Blackwell Publishing Ltd *Journal of Internal Medicine* 255: 105–112

## SOFTWARE CONSIDERATIONS

Regardless of the mode selected, the software must always issue the correct mode control word after a reset of the device. Generally, an initialization routine is constructed which issues the correct mode control word, sets up the initial state of the control lines, and initializes any program internal data.

Many of the software requirements of the 8255A vary as a function of the mode selected. The simplest mode supported by the device is Mode 0 (Basic Input/Output). Generally, Mode 0 is used for simple status driven device interfaces (no interrupts). Figure 10 illustrates sample software that could be used to support such interfaces. Most devices support a BUSY or READY signal which is used to determine when the device is ready to input or output data and a DATA STROBE which is used to request data transfer (DATA STROBE may easily be generated with the Port C bit set/reset feature). In the Mode 0 configuration, Ports A and B are used to input/output byte oriented data. Port C is used to input 8255A status, peripheral status and to drive peripheral control lines.

When the Mode 1 and Mode 2 configurations are used, the software is generally required to support interrupts. Software routines written for an interrupt driven environment tend to be more complex than status driven routines. The added complexity is due to the fact that interrupt driven systems are constructed such that other software tasks are run while the I/O transaction is in progress. A software routine that controls a peripheral device is generally referred to as a device driver. One method of implementing an interrupt driven device driver is to partition the device driver into a "Command Processor" and an "Interrupt Service Routine". The command processor is the module that validates

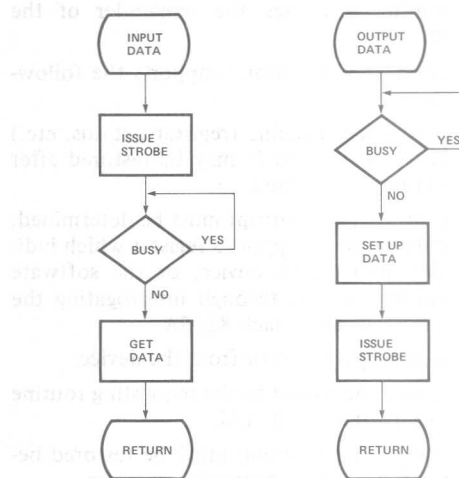


Figure 10. Sample Status Driven Software Flowchart

and initiates user program requests to the device driver. A common method of passing information between the various software programs is to have the requesting routine provide a device control block in memory. A sample device control block is shown in Table II.

Table II. Sample Device Control Block

NAME	DESCRIPTION
Status	This 1-byte field is used to transmit the status of the I/O transaction (busy, complete, etc.).
Opcode	This 1-byte field defines the type of I/O (READ, WRITE, etc.).
Buffer Address	This 2-byte field specifies the source/destination of the data block.
Character Count	This 1-byte field is a count of the number of characters involved in the transaction.
Character Transferred Count	This 1-byte count of the number of characters which were actually transferred.
Completion Address	This 2-byte field is the address of the user supplied completion routine which will be called after the I/O has been performed.

block. Control is then returned to the requestor so that other processing may proceed. The interrupt service routine processes the remainder of the transaction.

The interrupt service routine supports the following functions:

1. The state of the machine (registers, status, etc.) must be saved so that it may be restored after the interrupt is processed.
2. The source of the interrupt must be determined. The hardware may support a register which indicates the interrupting device, or the software may poll the devices through interrogating the Port C status word of each 8255A.
3. Data must be passed to or from the device.
4. Control must be passed to the requesting routine at the completion of the I/O.
5. The state of the machine must be restored before returning to the interrupted program.

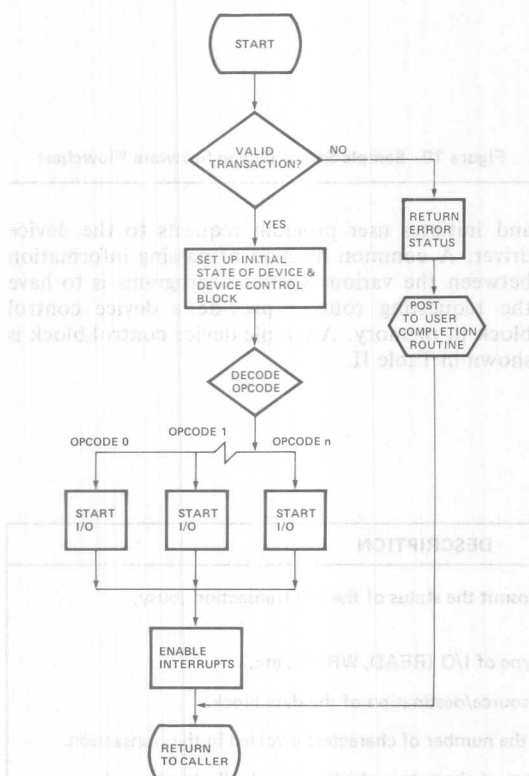


Figure 11. Command Processor

processor and interrupt service routine modules.

The rest of this application note presents specific application examples. All of the 8080 assembly language programs supplied with the application examples use the standard Intel 8080 assembly language mnemonics. The programs discussed use the program equate statement to specify all hardware related data. Equate statements are used so that all references to an I/O port may be changed through a simple reassignment of the port address in the equate statement.

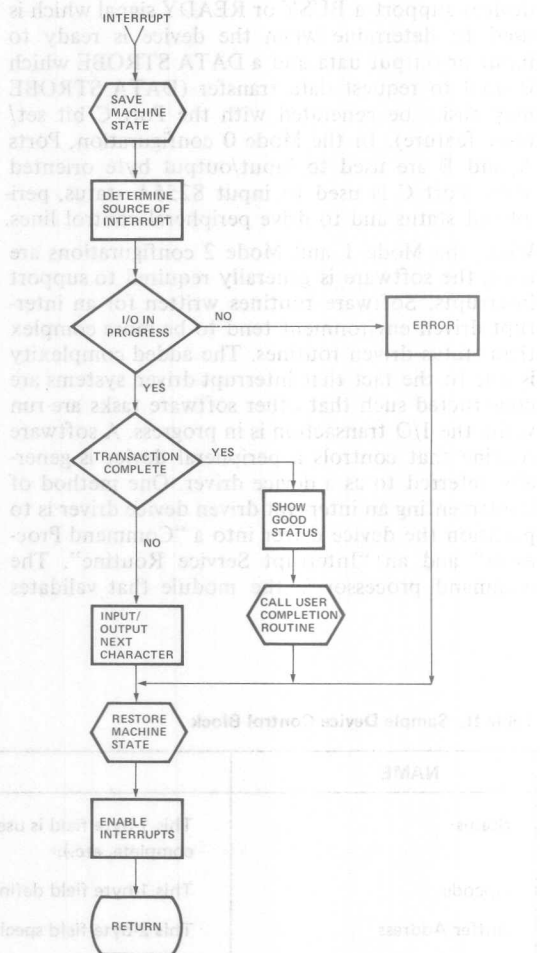


Figure 12. Interrupt Service Routine

## MODE 0 – STATUS DRIVEN PERIPHERAL INTERFACE

This design example shows how a single 8255A in Mode 0 may be used to develop a status driven interface (no interrupts) for the Centronics 306 character printer, the Remex paper tape punch, and the Remex paper tape reader.

### 8255A To Peripheral Hardware Interface

The first step in the design is to examine the specification for the peripheral devices and identify the control and data signals which must be supported by the interface. Table III lists the signals which were chosen to be supported by the 8255A interface. All three of the devices support the standard

Table III. Peripheral Interface Signals

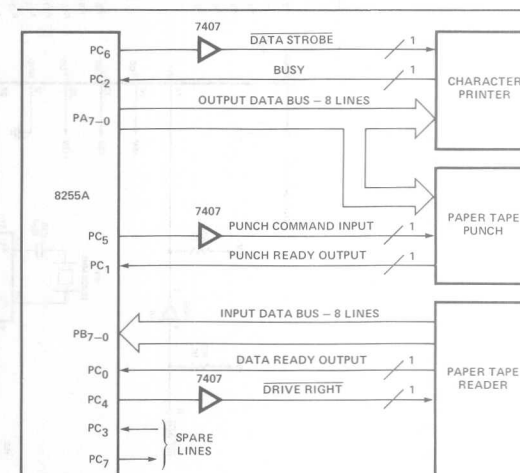
CHARACTER PRINTER	
Name:	DATA 0–DATA 7
Definition:	Input data levels. A high signal represents a binary 1 and a low signal represents a binary 0. These eight lines are the data lines to the printer.
Name:	DATA STROBE
Definition:	A 0.5 $\mu$ sec pulse (minimum) used to transfer data from the 8255A to the printer.
Name:	BUSY
Definition:	The level indicating that the printer cannot receive data.
PAPER TAPE PUNCH	
Name:	TRACKS 1–8 DATA INPUT
Definition:	Input data levels. A high signal causes a hole to be punched on the associated track. These eight lines are the data lines to the printer.
Name:	PUNCH COMMAND INPUT
Definition:	A true condition moves the tape and initiates punching the tape. This signal is actually a data strobe.
Name:	PUNCH READY OUTPUT
Definition:	True signal indicates that the punch is ready to accept a punch command. This is the punch busy line.
PAPER TAPE READER	
Name:	DATA TRACK OUTPUTS
Definition:	True signal indicates data track hole. These eight lines are the data lines from the punch.
Name:	DRIVE RIGHT
Definition:	True signal drives the tape to the right and reads a character. This signal is actually the data strobe (initiate read signal).
Name:	DATA READY OUTPUT
Definition:	True signal indicates data track outputs are in "On character" condition. This signal is the reader busy line.

BUSY/DATA STROBE interface discussed previously (see Figure 10). Figure 13 is a block diagram of the interface design. The 8255A Port A is configured as a Mode 0 output port which is used to support the printer and the paper tape punch data bus. Port B is configured as a Mode 0 input port and is used to input the paper tape reader data. Three of the Port C lower bits ( $PC_2$ – $PC_0$ ) configured in input mode are used to input the device busy indications. Three of the Port C upper bits ( $PC_6$ – $PC_4$ ) configured in output mode are used to support the device strobe signals required by each device.

The drive requirements of the interface lines are a function of the peripheral interface circuitry, the length of the interface cable, and the environment in which the unit is running. In this particular design example, all output lines from the 8255A to the peripherals were buffered through a 7407 buffer/driver. The input lines from the peripherals were fed directly into the Port C and Port B inputs.

### 8080 CPU Module To 8255A Interface

The schematic of the completed hardware design is shown in Figure 14. The CPU module design shown is the design which was implemented for Intel's SDK 80 kit board. The 8255A is addressed through the use of an isolated I/O architecture utilizing a linear select scheme. Address bits  $A_1$  and  $A_0$  are used to select the 8255A port. Address bit  $A_3$  is the exclusive enable for 8255A #1. Examination of the schematic shows that all of the 8255A interface lines are directly driven by the CPU module.



NOTE:  
1. OUTPUT DATA BUS BUFFERED WITH 7407.  
2. ALL 8255A OUTPUT LINES ARE PULLED UP TO +5V AT THE PERIPHERAL.

Figure 13. Interface Block Diagram

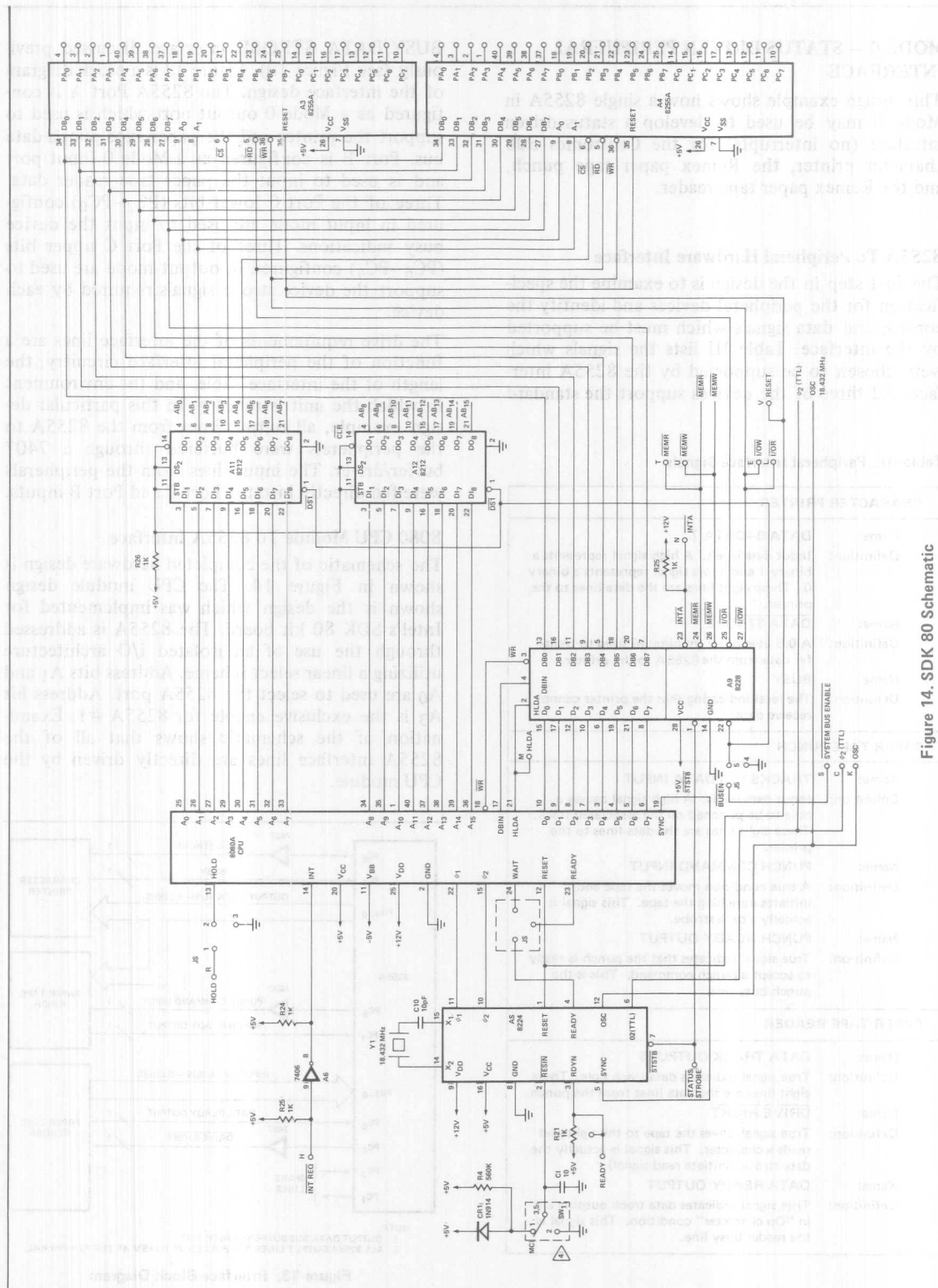


Figure 14. SDK 80 Schematic

## Mode 0 Interface Software

An initialization routine and three device drivers (one for each peripheral device) are required to support the peripheral interface. The I/O port addresses implemented by the hardware are shown in Figure 15. The unused chip select bits are set to one so that chip select conflicts will not result if the unused bits are required by an expanded system.

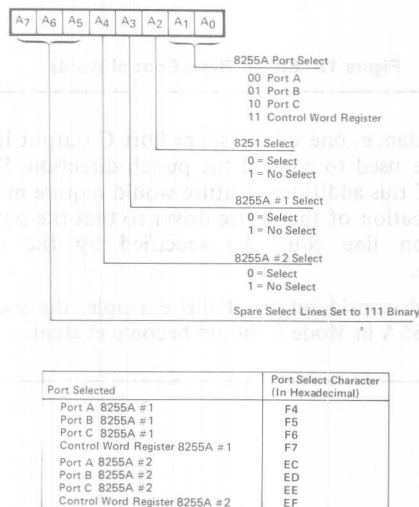


Figure 15. I/O Port Addresses

Note that the initialization routine issues the mode control word (shown in Figure 16). It also sets the low true DATA STROBE signals to an inactive (high) state.

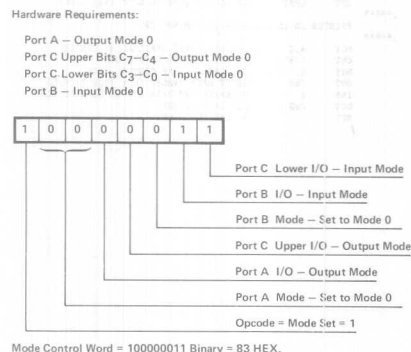


Figure 16. Mode Control Word

ISIS 8080 MACRO ASSEMBLER, V1.0  
MODE ZERO EXAMPLE

PAGE 1

```

*****
; TITLE "MODE ZERO EXAMPLE"
;
; CHARACTER PRINTER, PAPER TAPE PUNCH, PAPER TAPE READER
; MODE ZERO EXAMPLE
;
*****
; PROGRAM EQUATES
;
00F4 EQU 0F4H ; 8255 PORT A
00F5 EQU 0F5H ; 8255 PORT B
00F6 EQU 0F6H ; 8255 PORT C
00F7 EQU 0F7H ; 8255 CONTROL WORD REGISTER
;
; INITIALIZATION CONTROL WORD
;
; USED TO CONFIGURE THE 8255 AS FOLLOWS:
;
; PORT A - OUTPUT MODE ZERO
; PORT B - INPUT MODE ZERO
; PORT C (UPPER) - OUTPUT
; PORT C (LOWER) - INPUT
;
0083 EQU 10000011B ; INITIALIZATION CONTROL WORD
;
; SET/RESET CONTROL WORDS FOR GENERATION OF DATA STROBES
; ON PORT C.
;
000D EQU 00001101B ; PRINTER DATA STROBE ON
000C EQU 00001100B ; PRINTER DATA STROBE OFF
000B EQU 00001011B ; PUNCH DATA STROBE ON
000A EQU 00001010B ; PUNCH DATA STROBE OFF
0009 EQU 00001001B ; READER DATA STROBE ON
0008 EQU 00001000B ; READER DATA STROBE OFF
;
; BIT MASK FOR DEVICE BUSY CHECK
;
0004 EQU 04H ; LINE PRINTER BUSY
0002 EQU 02H ; PUNCH BUSY
0001 EQU 01H ; READER BUSY

```

ISIS 8080 MACRO ASSEMBLER, V1.0  
MODE ZERO EXAMPLE - INITIALIZATION ROUTINE

PAGE 2

```

*****
; PROGRAM ORIGIN
;
3000 ORG 03000H
;
; INITIALIZATION ROUTINE
; A REGISTER MODIFIED
;
INIT:
3000 3B83 MVI A,ICW ; GET INITIALIZATION CONTROL WORD
3002 D3F7 OUT CWR ; OUTPUT TO CONTROL WORD REGISTER
;
; SET ALL LOW TRUE DATA STROBES ON
;
3004 3E0D MVI A,LPSON ; GET CONTROL WORD TO TURN ON PRINTER DATA STROBE
3006 D3F7 OUT CWR ; OUTPUT TO CONTROL WORD REGISTER
3008 3E09 MVI A,RDSON ; GET CONTROL WORD TO TURN ON READER DATA STROBE
300A D3F7 OUT CWR ; OUTPUT TO CONTROL WORD REGISTER
300C C9 RET ; RETURN TO CALLER

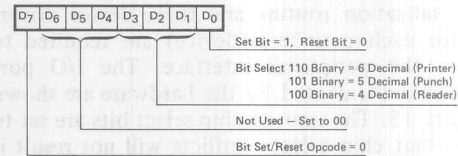
```

The three peripheral drivers which follow all have the basic structure discussed previously. Consider the printer routine. Here the user routine places an ASCII data character in the C-register and passes control to the LPST location through a subroutine call. The printer driver interrogates the status of the printer by reading Port C. If the printer is busy, the routine will loop until the printer is idle. When the printer is ready to accept a data character, the character is placed on the Port A lines and a DATA STROBE is generated. After generating the DATA STROBE, the driver executes a subroutine return to the caller.

The DATA STROBE signals to the devices are generated through the use of the Port C bit set/reset feature. The bit set/reset control words used are shown in Figure 17.

### Summary/Conclusions

This design example discussed the basic hardware and software required to handle a simple device interface. The 8255A will easily accommodate a more complex interface design which utilizes additional interface lines supported by the peripheral.

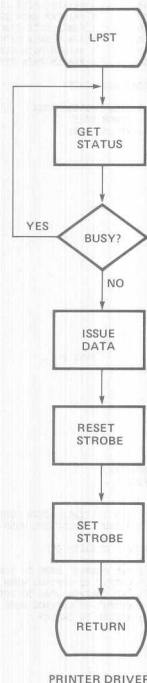


The control word for set Printer DATA STROBE (PC 6) = 00001101 binary.  
The control word for reset Printer DATA STROBE (PC 6) = 00001100 binary.  
The control word for set Punch DATA STROBE (PC 5) = 00001011 binary.  
The control word for reset Punch DATA STROBE (PC 5) = 00001010 binary.  
The control word for set Reader DATA STROBE (PC 4) = 00001001 binary.  
The control word for reset Reader DATA STROBE (PC 4) = 00001000 binary.

Figure 17. Bit Set/Reset Control Words

For instance, one of the spare Port C output lines may be used to control the punch direction. Support of this additional feature would require minor modification of the device driver so that the punch direction line could be specified by the user routine.

Through consideration of this example, the use of the 8255A in Mode 0 should become evident.



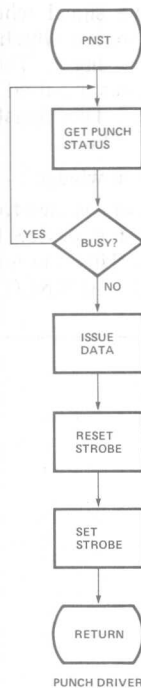
ISIS 8080 MACRO ASSEMBLER, V1.0  
MODE ZERO EXAMPLE - CHARACTER PRINTER DRIVER

PAGE 3

```

;*****
; CHARACTER PRINTER DRIVER
;
; INPUTS : CHARACTER TO PRINT IN C-REG
; OUTPUTS: CHARACTER TO PRINTER
;
; A REGISTER MODIFIED
;*****
LPST:
IN    PORTC ; GET STATUS OF PRINTER
ANI   LPBSY ; SEE IF BUSY
JNZ   LPST  ; IF BUSY - JUMP TO LPST (WAIT LOOP)
;*****
; PRINTER IS IDLE - OUTPUT A CHARACTER
;*****
MOV    A,C  ; GET DATA BYTE SUPPLIED BY CALLER
OUT    PORTA ; OUTPUT DATA TO DATA LINES
MVI    A,LPSTOF ; GET DATA STROBE CONTROL WORD
OUT    CWR  ; RESET DATA STROBE (LOW TRUE SIGNAL)
INR    A    ; GENERATE SET DATA STROBE CONTROL WORD
OUT    CWR  ; SET DATA STROBE
RET

```



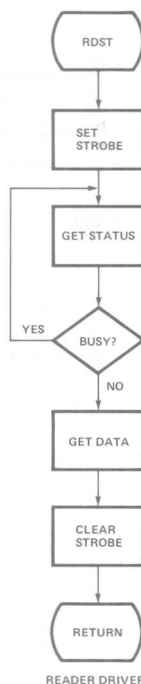
ISIS 8080 MACRO ASSEMBLER, V1.0  
MODE ZERO EXAMPLE - PAPER TAPE PUNCH DRIVER

PAGE 4

```

*****
;
; PAPER TAPE PUNCH DRIVER
;
; INPUTS : DATA TO PUNCH IN C-REGISTER
; OUTPUTS: DATA TO PUNCH
;
; A REGISTER MODIFIED
;
*****
PNST:
;
; IN   PORTC ; GET STATUS OF PUNCH
; ANI  PNBYS ; SEE IF BUSY
; JNZ  PNST  ; IF BUSY - JUMP TO PNST (WAIT LOOP)
;
; PUNCH IS IDLE - OUTPUT A CHARACTER
;
*****
3026 79      MOV   A,C ; GET DATA BYTE SUPPLIED BY CALLER
;          OUT   PORTA ; OUTPUT DATA TO DATA LINES
3029 3E08    MVI   A,PNSON ; SET DATA STROBE CONTROL WORD
302B D3F7    OUT   CWR ; SET DATA STROBE
302D 3D      DCR   A ; GENERATE RESET DATA STROBE CONTROL WORD
302E D3F7    OUT   CWR ; RESET DATA STROBE
3030 C9      RET   ; RETURN TO CALLER

```



ISIS 8080 MACRO ASSEMBLER, V1.0  
MODE ZERO EXAMPLE - PAPER TAPE READER DRIVER

PAGE 5

```

*****
;
; PAPER TAPE READER DRIVER
;
; INPUTS : DATA FROM READER
; OUTPUTS: CHARACTER TO USER IN C-REGISTER
;
; A AND C REGISTER MODIFIED
;
*****
RDST:
;
; MVI  A,RDSOF ; GET STROBE CONTROL WORD (LOW TRUE SIGNAL)
; OUT  CWR     ; SET DATA STROBE
;
; RDLP:
;
; IN   PORTC ; GET STATUS OF DEVICE
; ANI  RDBYS ; SEE IF BUSY
; JNZ  RDLP  ; IF BUSY - LOOP UNTIL IDLE
;
; READER NOT BUSY - GET CHAR AND CLEAR STROBE
;
*****
;
; IN   PORTB ; GET CHARACTER
; MVI  C,A   ; SAVE CHARACTER
3040 3E09    MVI  A,RDSON ; GET STROBE SET CONTROL WORD (LOW TRUE SIGNAL)
3042 D3F7    OUT  CWR ; TURN OFF STROBE
3044 C9      RET   ; RETURN TO CALLER
;
; *****
;
; END OF MODE ZERO EXAMPLE
;
*****
0000      END

```

## MODE 1 INTERRUPT DRIVEN PRINTER INTERFACE

The status driven interface described in the previous example required the software driver to poll the device status for completion. An alternate approach is to construct the device interface such that an interrupt is used to signal the completion of the operation. When an interrupt driven interface is utilized, the time that was dedicated to polling can be used to perform other functions and the effective processor through-put is increased. This example demonstrates how an 8255A configured in Mode 1 may be used to develop an interrupt driven interface for the Centronics 306 character printer.

### CPU Module To 8255A Interface

The 8080 bus interface implemented for this example is the same as the Mode 0 example with the addition of interrupt support. Interrupt support is implemented through the use of a special feature of the 8228 System Controller. If only one interrupt vector is required (such as in small systems), the 8228 can automatically assert an RST 7 instruction onto the data bus at the proper time. This option is selected by connecting the  $\overline{\text{INTA}}$  output of the 8228 to the +12-volt supply through a 1K ohm series resistor.

The Mode 1 interrupt support logic of the 8255A provides an interrupt request line for each port. The 8255A interrupt request line ( $\text{INTRA}$ ) must be connected to the INT line of the 8080. A 10K ohm pullup resistor is used to insure that the  $V_{\text{IH}}$  requirements of the 8080 are met.

### 8255A To Peripheral Interface

The interrupt driven configuration control signal interface to the printer is different than the status driven interface. Instead of a BUSY/DATA STROBE interface, a DATA STROBE/ACK interface is supported. The ACK signal notifies the 8255A that a character transferred to the printer by a DATA STROBE has been accepted. After an ACK is issued the printer is considered idle. The block diagram shown in Figure 18 displays the interface signals used.

The Mode 1 interrupt driven peripheral support signals used are:

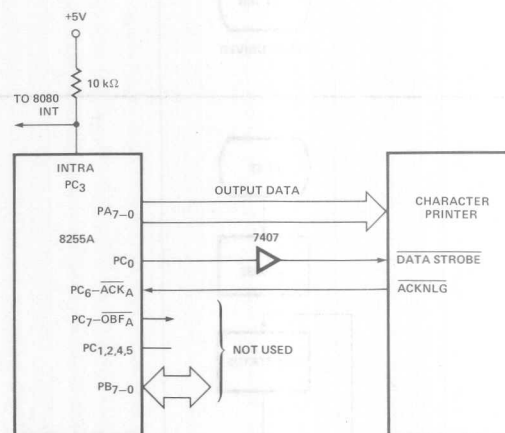
- $\text{PA}_7\text{--PA}_0$  — Output Data  
Used to support the printer data port.
- $\overline{\text{OBF}}$  — Output Buffer Full  
This line goes low when data is placed in the output buffer. The OBF signal may be used as a data

strobe signal when interfacing to peripherals which do not require a pulsed input. The Centronics 306 requires a pulsed DATA STROBE signal. This signal is supported by Port C bit 0.

$\overline{\text{ACK}}$

— Acknowledge

This line is used to signal the 8255A that the device has accepted the data. This line is supported by the printer ACKNLG signal.



NOTES:

1. DATA BUS BUFFERED WITH 7407.
2. ALL 8255A OUTPUT LINES ARE PULLED UP TO +5V AT THE PERIPHERAL.

Figure 18. Interface Block Diagram

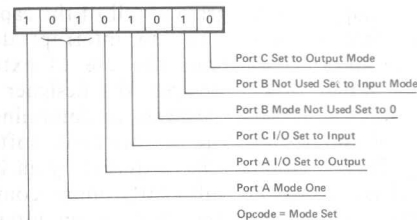
## Mode 1 Software Driver

The software driver implemented for this example utilizes the typical interrupt driven software structure outlined previously. The initialization routine issues the mode control word (shown in Figure 19) to the 8255A after reset of the device. The initialization routine also places a jump to the interrupt service routine in the interrupt location for RST 7. The command processor is started by the user routine through a subroutine call to PSTRT, with the address of the control block in the D and E registers (the control block format is shown in Table IV). The command processor insures that an I/O operation is not already in progress, starts the I/O, enables interrupts, and returns to the caller so that other processing may proceed.

After a character is placed in the output buffer, the DATA STROBE signal is generated through the use of the Port C bit set/reset feature. When the ACK is generated by the printer, the buffer full indication is cleared and the 8255A generates an interrupt. If interrupts are enabled, the interrupt request is serviced by the 8080 CPU through disabling processor interrupts and then executing the instruction at location 38 hexadecimal in program memory. The interrupt service routine saves the processor state and polls the 8255A to determine the source of the interrupt. Once the interrupting device is located, the control block is used to locate the next data character for transfer to the 8255A output buffer. After the entire buffer has been printed, the interrupt service routine passes control to the user-supplied completion routine. Before returning from the interrupt, the state of the processor is restored.

### Hardware Requirements:

Port A — Input Mode 1  
Port C (Lower) — Output  
Port B  
Port C (Upper) — Not Used



Mode Control Word = 10101010 Binary = AA HEX.

Figure 19. Mode Control Word

Table IV. Printer Software Control Block

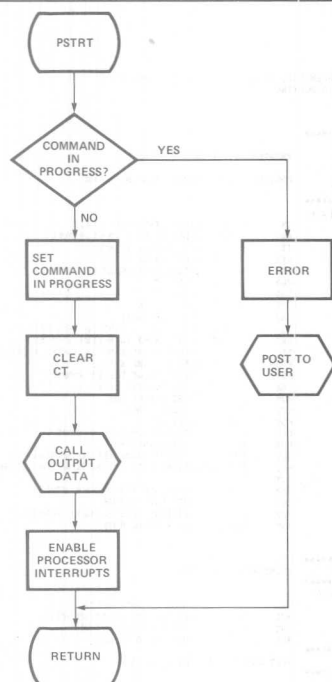
NAME	POSITION	DEFINITION
Status	Byte 0	A 1-byte field which defines the completion status of an I/O. 00 = Good completion 01 = Error — command already in progress
Buffer Address	Byte 1, 2	Pointer to the start of the data to print.
Character Count	Byte 3	Count of the number of characters to print.
Character Transferred Count	Byte 4	The number of characters transferred.
Completion Address	Byte 5, 6	Address of a user supplied routine which will be called after the I/O has been performed.

### NOTES:

1. An opcode field is not required because WRITE is the only operation performed.
2. The control block must reside above location FF Hex.

... -- P-0-0-0-0, or all

ISIS 8080 MACRO ASSEMBLER, V1.0 PAGE 2  
MODE ONE EXAMPLE

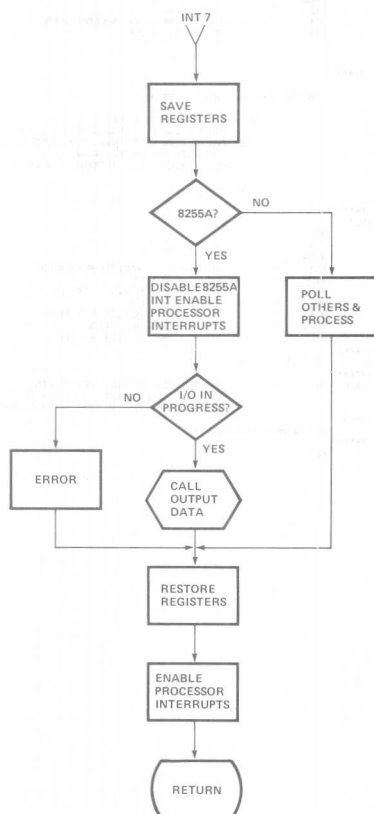


ISIS 8080 MACRO ASSEMBLER, V1.0  
COMMAND PROCESSOR

PAGE 3

```

;*****
; COMMAND PROCESSOR
; INPUTS: CONTROL BLOCK ADDRESS IN D AND E REGISTERS
; OUTPUTS: START I/O OR ERROR STATUS IN CONTROL BLOCK
; A,H,L REGISTERS MODIFIED
;*****
PSTRT:
3014 3AA230 LDA PIPRG+1 ; GET PRINT IN PROGRESS BLOCK ADDRESS
3017 A7 ANA A ; SEE IF ZERO (PRINT IN PROGRESS)
; IF BLOCK ADDRESS NOT EQUAL TO ZERO THEN
; PRINT IN PROGRESS
; IF YES - BRANCH TO ERROR
3018 C2B330 JNZ PSTE ; SAVE CONTROL BLOCK ADDRESS
3019 EB XCHG
301C 22A130 SHLD PIPRG
301F EB XCHG
3020 210400 LXI H,CBCT ; GET INDEX TO CT
3023 19 DAD D ; COMPUTE ADDRESS OF CT
3024 3600 MVI M,DOH ; CLEAR CT
3026 CD5830 CALL PDATA ; START I/O
3029 FB EI ; ENABLE PROCESSOR INTERRUPTS
302A C9 RET ; RETURN TO CALLER
;*****
; ERROR - TRANSACTION ALREADY IN PROGRESS
;*****
PSTE:
302B 3B01 MVI A,STE1 ; GET ERROR STATUS CODE
302D C39430 JMP POST ; PASS CONTROL TO COMPLETION ROUTINE
  
```

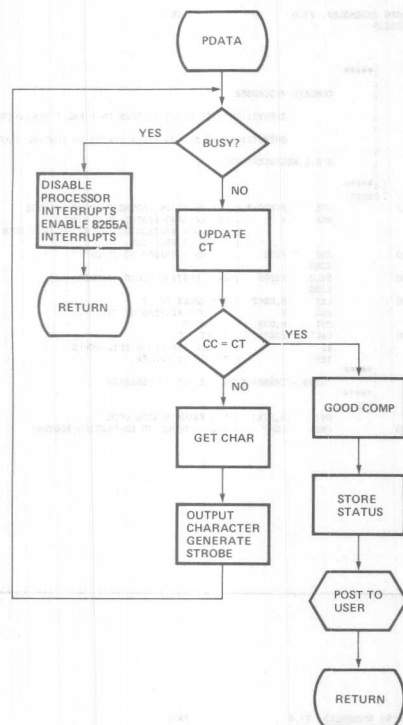


ISIS 8080 MACRO ASSEMBLER, V1.0  
PRINTER INTERRUPT SERVICE ROUTINE

PAGE 4

```

;*****
; PRINTER INTERRUPT SERVICE ROUTINE
; ALL REGISTERS SAVED AND RESTORED
;*****
PINT:
3030 F5 PUSH PSW ; SAVE PSW
3031 C5 PUSH B ; SAVE REGISTER PAIR B AND C
3032 D5 PUSH D ; SAVE REGISTER PAIR D AND E
;*****
; POLL INTERRUPT SOURCE - SEE IF 8255
;*****
3034 D8F6 IN PORTC ; GET STATUS OF DEVICE
3036 E608 ANI INTA ; SEE IF INT
3038 CA5230 JZ PPOLL ; NO - BRANCH TO POLL OTHER DEVICES IF ANY
303B 3B0C MVI A,IDN ; GET 8255 INT DISABLE CONTROL WORD
303D D3F7 OUT CWR ; DISABLE DEVICE INTERRUPTS
303F FB EI ; ENABLE PROCESSOR INTERRUPTS
3040 2AA130 LHLD PIPRG ; GET CONTROL BLOCK ADDRESS
3043 AF XRA A ; CLEAR A REG
3044 BC CMP H ; SEE IF PRINT IN PROGRESS
3045 CA5530 JZ PIER1 ; NO - BRANCH TO ERROR ROUTINE
3048 EB XCHG
3049 CD5830 CALL PDATA ; PRINT DATA
;*****
; RESTORE REGISTERS AND RETURN FROM INTERRUPT
;*****
PRTN:
304C E1 POP H ; RESTORE REGISTER PAIR H AND L
304D D1 POP D ; RESTORE REGISTER PAIR D AND E
304E C1 POP B ; RESTORE REGISTER PAIR B AND C
304F F1 POP PSW ; RESTORE PSW
3050 FB EI ; ENABLE PROCESSOR INTERRUPTS
3051 C9 RET ; RETURN TO INTERRUPTED PROCESS
;*****
; POLL OTHER DEVICES IF ANY
; IF NO OTHER DEVICES TO POLL - USER SUPPLIED ERROR
; RECOVERY ROUTINE.
;*****
PPOLL:
3052 C3AC30 JMP PRTN ; RETURN
;*****
; ERROR - INTERRUPT FROM IDLE DEVICE
; USER SUPPLIED ERROR RECOVERY ROUTINE
;*****
PIER1:
3055 C3AC30 JMP PRTN ; RETURN
  
```



```

*****
: PRINTER OUTPUT DATA ROUTINE
: CONTROL BLOCK ADDRESS IN D AND E REG
*****
PDATA:
3058 DBF6 IN PORTC ; GET STATUS OF DEVICE
305A B5B0 ANI LPSST ; SEE IF BUSY (BUFFER FULL)
305C C8A30 JZ PDIO ; IF BUSY - BRANCH
305F 210400 LXI H,CBCT ; GET INDEX TO CT
3062 19 DAD D ; COMPUTE ADDRESS OF CT
3063 7E MOV A,M ; GET CT
3064 34 INR M ; INC CT
3065 2B DCX H ; DEC TO CC
3066 BE CMP M ; SEE IF EQUAL
3067 C8A30 JZ PCOMP ; IF EQUAL - DONE GO TELL USER
306A 210100 LXI H,CBUP ; GET INDEX TO BUFFER ADDRESS
306D 19 DAD D ; COMPUTE ADDRESS OF BUFFER ADDRESS
306E 05 PUSH D ; SAVE D AND E REGISTERS
306F 5E MOV E,M ; GET LSB OF BUFFER ADDRESS
3070 23 INX H ; INC TO NEXT BYTE
3071 56 MOV D,M ; GET BUFFER MSB
3072 2600 MVI H,00H ; CLEAR H REG
3074 6F MOV L,A ; GET CT
3075 19 DAD D ; COMPUTE CHARACTER ADDRESS
3076 7E MOV A,M ; GET CHARACTER
3077 D3F4 OUT PORTA ; OUTPUT CHARACTER TO PRINTER
3079 3800 MVI A,STBOF ; RESET DATA STROBE (LOW TRUE SIGNAL)
307B D3F7 OUT CWR ; GENERATE SET CONTROL WORD
307D 3C INR A ; SET DATA STROBE
307E D3F7 OUT CWR ; SET DATA STROBE
3080 D1 POP D ; RESTORE CONTROL BLOCK ADDRESS
3081 C35830 JMP PDATA ; LOOP UNTIL BUSY

```

\*\*\*\*\*  
PRINTER BUSY - RETURN

```

PDIO:
3084 F3 DI ; DISABLE INTERRUPTS
3085 3E00 MVI A,IEN ; ENABLE DEVICE INTERRUPTS
3087 D3F7 OUT CWR ; SET INTERRUPT ENABLE
3089 C9 RET ; RETURN TO CALLER

```

\*\*\*\*\*  
POST GOOD COMPLETION TO USER

```

PCOMP:
308A 3E00 MVI A,STOD ; GET GOOD STATUS CODE
308F AF CALL POST ; POST TO USER
3090 32A230 XRA A ; CLEAR A REG
3093 C9 RET ; RETURN TO CALLER

```

\*\*\*\*\*  
POST TO USER COMPLETION ROUTINE

```

: INPUTS : STATUS CODE IN A REG
: CONTROL BLOCK ADDRESS IN D AND E REG
: OUTPUTS: PASSES CONTROL TO USER COMPLETION ADDRESS
: SPECIFIED IN CONTROL BLOCK
: WITH CONTROL BLOCK ADDRESS IN D AND E
:
: A,H,L,B,C REG MODIFIED

```

```

POST:
3094 EB XCHG ;
3095 77 MOV M,A ; UPDATE STATUS
3096 EB XCHG ;
3097 210500 LXI H,CBCHP ; GET INDEX TO COMPLETION ADDRESS
309A 19 DAD D ; COMPUTE ADDRESS
309B 4E MOV C,M ; GET LSB OF COMPLETION ADDRESS
309C 23 INX H ; INC TO NEXT BYTE
309D 46 MOV B,M ; GET MSB OF COMPLETION ADDRESS
309E C5 PUSH B ; PUSH ADDRESS INTO STACK
309F C9 RET ; PASS CONTROL TO USER ROUTINE

```

\*\*\*\*\*  
DATA AND TABLES

```

30A1 0000 PIPRG: DW 0 ; IN PROGRESS CONTROL BLOCK ADDRESS
: IF DATA = 0 NO CONTROL BLOCK IN PROGRESS
: IF DATA NOT EQUAL TO ZERO CONTROL BLOCK IN PROGRESS

```

\*\*\*\*\*  
END OF MODE ONE EXAMPLE

\*\*\*\*\*  
END

## MODE 2 – 8080 TO 8080 INTERFACE

Due to the drastic reduction of hardware costs, system designs which utilize multiple CPU Modules are becoming more common. An 8080 may be configured as a master CPU and used to control multiple 8080 slave modules which act as intelligent I/O controllers. When multiple CPUs are utilized, a method of processor intercommunication must be supported. Figure 20 is a block diagram of one method of implementing a master/slave interface through the use of the 8255A Mode 2 bidirectional bus.

### Hardware Discussion

Two complete 8080 systems are required for this example. Intel's SBC 80/10 OEM board is used as the master CPU module and Intel's SDK 80 board is used as the slave CPU. The SBC 80/10 supports an 8255A which is configured in Mode 2. The 8255A is selected through the use of a decoded select scheme. Through the use of the 8228 RST 7 interrupt feature, a simple interrupt structure is supported. The SDK 80 is configured without interrupts for this example. The external logic required for this example is associated with the slave CPU. Simple logic is implemented which allows the slave CPU to generate the ACK and STB signals required to READ from and WRITE to the 8255A bidirectional bus with a single I/O instruction.

The system shown in Figure 20 utilizes SSI logic to read the 8255A IBF and OBF signals. If two spare 8255A input lines are available they could be used to input the IBF and OBF signals and eliminate the SSI logic.

### Software Discussion

Two sets of software are required to support the processor to processor interface. The master resident software which follows conforms to the simple interrupt driven software structure outlined previously. The initialization routine issues the Mode 2 control word to the 8255A after device reset. The command processor accepts READ/WRITE control blocks which provide a simple user interface for transferring data to/from the slave CPU. The master software is capable of processing both a read and a write control block simultaneously. The slave resident software shown at the end of this example utilizes the status driven approach.

### Summary/Conclusions

It is important to note that this design may be expanded to include more slave CPUs by simply adding another 8255A to the master module for each slave. The software drivers discussed address only the passing of data between the two processors. Specific applications generally dictate a software protocol be implemented for information transfer.

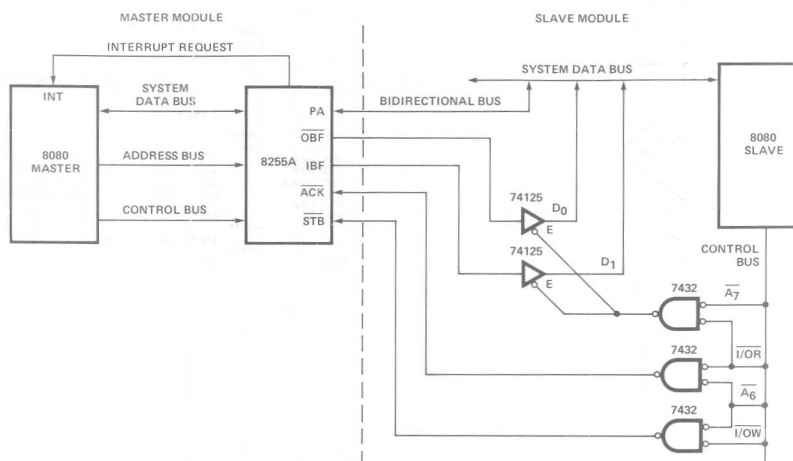
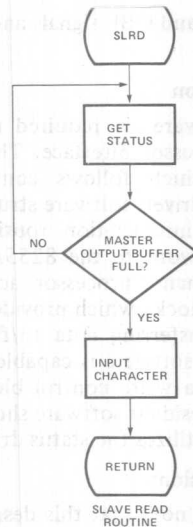
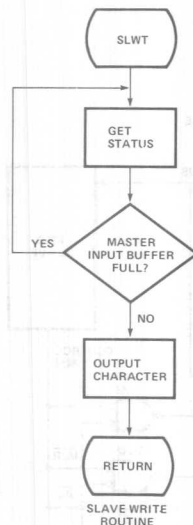


Figure 20. Interface Block Diagram



```

*****
TITLE 'MODE TWO EXAMPLE - SLAVE SOFTWARE'
*****
8080 MASTER TO 8080 SLAVE INTERFACE
- SLAVE SOFTWARE -
MODE TWO EXAMPLE
*****
*****
PROGRAM EQUATES
*****
00BF EQU 0BFH ; INTERPROCESSOR DATA PORT
007F EQU 07FH ; STATUS
*****
*****
BUFFER STATUS MASKS
*****
0001 EQU 01H ; OUTPUT BUFFER FULL
0002 EQU 02H ; INPUT BUFFER FULL
*****
*****
PROGRAM ORIGIN
*****
3000 ORG 03000H
*****
*****
SLAVE READ ROUTINE
*****
INPUTS: NONE
OUTPUTS: CHARACTER READ IN C-REGISTER
A,C REG MODIFIED
*****
SLRD:
3000 DB7F IN PSTS ; GET STATUS
3002 E601 ANI 0BF ; SEE IF BUFFER FULL
3004 C20030 JNZ SLRD ; NO - LOOP UNTIL FULL
3007 DB8F IN PDATA ; GET CHARACTER
3009 AF MOV C,A ; PLACE IN C-REG
300A C9 RET ; RETURN TO CALLER
  
```



```

*****
*****
SLAVE WRITE ROUTINE
*****
INPUTS: CHARACTER TO WRITE IN C-REGISTER
OUTPUTS: NONE
A REG MODIFIED
*****
*****
SLWT:
300B DB7F IN PSTS ; GET STATUS
300D E602 ANI 0BF ; SEE IF BUFFER FULL
300F C20B30 JNZ SLWT ; YES - LOOP UNTIL EMPTY
3012 79 MOV A,C ; GET DATA CHARACTER
3013 DB8F OUT PDATA ; OUTPUT DATA
3015 C9 RET ; RETURN TO CALLER
*****
*****
END OF SLAVE SOFTWARE DRIVER
*****
0000 END
  
```

```

;*****
; TITLE 'MODE TWO EXAMPLE - MASTER SOFTWARE'
;*****
;
; 8080 MASTER TO 8080 SLAVE INTERFACE
;   MASTER SOFTWARE
;   MODE TWO EXAMPLE
;*****
;*****
;***** PROGRAM EQUATES
;*****
00E4 PORTA EQU 0E4H ; 8255 PORT A
00E5 PORTB EQU 0E5H ; 8255 PORT B
00E6 PORTC EQU 0E6H ; 8255 PORT C
00E7 CWR EQU 0E7H ; 8255 CONTROL WORD REGISTER
0038 RST7 EQU 038H ; RESTART 7 ADDRESS
;*****
;
; INITIALIZATION CONTROL WORD
;
; USED TO CONFIGURE THE 8255 AS FOLLOWS:
;
; PORT A - MODE 2 BIDIRECTIONAL BUS
; PORT B - INPUT MODE 0 (NOT USED)
; REMAINING PORT C LINES - INPUT MODE (NOT USED)
;*****
00CB ICW EQU 11001011B ; INITIALIZATION CONTROL WORD
;*****
;***** 8255 ENABLE/DISABLE INTERRUPT CONTROL WORDS
;*****
000D IENI EQU 00001101B ; ENABLE INPUT INTERRUPTS
0009 IENO EQU 00001001B ; ENABLE OUTPUT INTERRUPTS
000C IDNI EQU 00001100B ; DISABLE INPUT INTERRUPTS
000B IDNO EQU 00001000B ; DISABLE OUTPUT INTERRUPTS
;*****
;***** STATUS EQUATES
;*****
0008 INTRA EQU 08H ; INTERRUPT REQUEST
0080 OBFA EQU 80H ; OUTPUT BUFFER FULL
0020 IBFA EQU 20H ; INPUT BUFFER FULL

```

```

;*****
;***** CONTROL BLOCK EQUATES
;*****
0000 CBST EQU 00H ; STATUS BYTE
; CBOP EQU 01H ; OPCODE = 0 READ
; ; OPCODE = 1 WRITE
0002 CBUF EQU 02H ; BUFFER ADDRESS
0004 CBCC EQU 04H ; CHARACTER COUNT
0005 CBCT EQU 05H ; CHARACTER TRANSFERRED COUNT
0006 CBCKP EQU 06H ; COMPLETION SERVICE ADDRESS
;*****
;***** OPCODE EQUATES
;*****
0000 OPRD EQU 00H ; READ OPCODE
0001 OPWT EQU 01H ; WRITE OPCODE
;*****
;***** COMPLETION STATUS EQUATES
;*****
0000 STGD EQU 00H ; GOOD COMPLETION
0001 STE1 EQU 01H ; ERROR - COMMAND ALREADY IN PROGRESS
0002 STE2 EQU 02H ; ERROR - INVALID OPCODE
;*****
;***** SET UP INTERRUPT VECTOR
;*****
0038 ORG RST7
0038 C34630 JMP PINT ; JUMP TO INTERRUPT SERVICE ROUTINE
;*****
;***** PROGRAM ORIGIN
;*****
3000 ORG 03000H
;*****
;*****
;***** INITIALIZATION ROUTINE
;*****
;***** A REGISTER MODIFIED
;*****
INIT:
3000 3ECB MVI A,ICW ; GET MODE CONTROL WORD
3002 D3E7 OUT CWR ; OUTPUT TO CONTROL WORD REGISTER
3004 C9 RET ; RETURN TO CALLER

```



ISIS 8080 MACRO ASSEMBLER, V1.0  
COMMAND PROCESSOR

PAGE 3

```

:
:
:      COMMAND PROCESSOR
:
:      INPUTS: CONTROL BLOCK ADDRESS IN D AND E REGISTERS
:
:      OUTPUTS: START I/O OR ERROR STATUS IN CONTROL BLOCK
:      MODIFIED
:
:      A,H,L
:
:
: *****
: PSTRT:
3005 210500      LXI   H,CBCT ; GET INDEX TO CT
3008 19         DAD   D      ; COMPUTE ADDRESS OF CT
3009 3600      MVI   M,OPRD ; CLEAR CT
3010 210100      LXI   H,CBOP ; GET INDEX TO OPCODE
3012 19         DAD   D      ; COMPUTE ADDRESS
3013 19         MOV   A,M     ; GET OPCODE
3010 FE00      CPI   00H     ; SEE IF READ
3012 CA2430      JZ   PSRD   ; YES - GO PROCESS READ
3015 FE01      CPWT   00H     ; SEE IF WRITE
3017 CA3530      JZ   PSWT   ; YES - GO PROCESS WRITE
:
: *****
:      ERROR - INVALID OPCODE
:
: *****
301A 3E02      MVI   A,STE2 ; GET ERROR STATUS CODE
:      JMP   POST ; CALL COMPLETION ROUTINE
:
: *****
:      ERROR - TRANSACTION ALREADY IN PROGRESS
:
: *****
: PSTE:
301F 3E01      MVI   A,STE1 ; GET ERROR STATUS CODE
3021 C3DC30      JMP   POST ; CALL COMPLETION ROUTINE
:
: *****
:      PROCESS READ COMMAND
:
: *****
: PSRD:
3024 3A6A30      LDA   PRGRD+1 ; GET READ IN PROGRESS ADDRESS
3027 A7         ANA   A      ; SEE IF READ IN PROGRESS (TEST FOR ZERO)
3028 C21F30      JNZ   PSTE  ; IF YES - BRANCH
3028 B8         XCHG
302C 2B6930      SHLD  PRGRD ; SAVE CONTROL BLOCK ADDRESS
302F E6         XCHG
3030 C07C30      CALL  PIN   ; START I/O
3033 F8         EI         ; ENABLE INTERRUPTS
3034 C9         RET        ; RETURN TO CALLER

```

ISIS 8080 MACRO ASSEMBLER, V1.0  
COMMAND PROCESSOR

PAGE 4

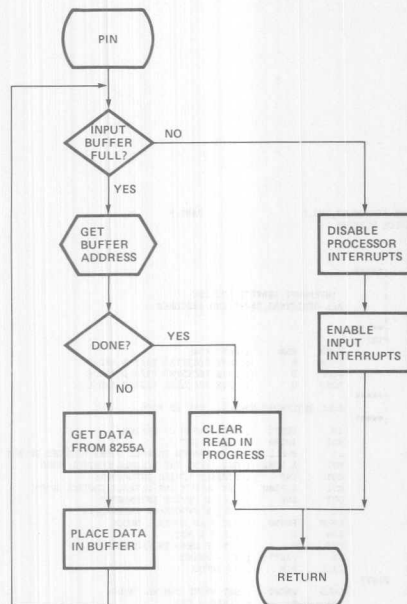
```

: *****
:
: *****
:
PSMT:
PROCESS WRITE COMMAND

3035 3AEC30 LDA PRGWT-1 : GET WRITE IN PROGRESS ADDRESS
3036 A7 ANA A : SEE IF WRITE IN PROGRESS (TEST FOR ZERO)
3039 C21F30 JNZ PSTE : IF YES - BRANCH
303C EB XCHG XCHG
303D 22EB30 SHLD PRGWT : SAVE CONTROL BLOCK ADDRESS
3040 EB XCHG XCHG
3041 CD9C30 CALL POUT : START I/O
3044 F1 EI : ENABLE INTERRUPTS
3045 C9 RET : RETURN TO CALLER

```





ISIS 8080 MACRO ASSEMBLER, V1.0  
INPUT DATA ROUTINE

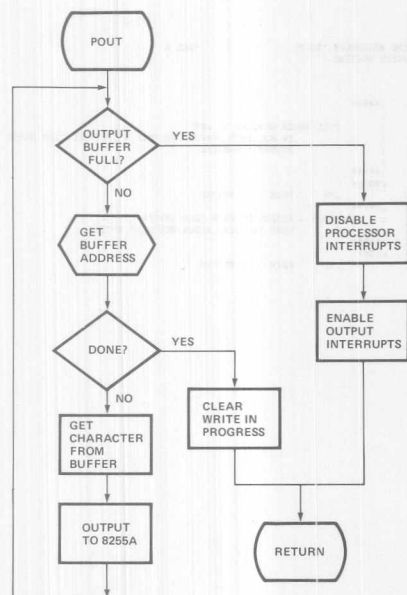
PAGE 7

```

;*****
;***** INPUT DATA ROUTINE
;*****
PIN:
307C DBE6      IN      PORTC ; GET STATUS OF DEVICE
307E E620      ANI     IBFA  ; SEE IF INPUT BUFFER FULL
3080 CA9630    JZ      PRTI  ; NO - BRANCH
3083 CDBC30    CALL    CBFA  ; GET ADDRESS IN BUFFER
3086 DA8F30    JC      PIDON  ; IF DONE - BRANCH
3089 DBE4      IN      PORTA ; GET DATA
308B 77        MOV     M,A    ; PLACE IN BUFFER
308C C37C30    JMP     PIN    ; LOOP

;*****
;***** END OF INPUT TRANSACTION
;*****
PIDON:
308F AF        XRA      A      ; CLEAR A
3090 32BA30    STA     PRGRD+1 ; CLEAR READ IN PROGRESS
3093 C39630    JMP     PRTI    ; RETURN

;*****
;***** RETURN FROM INPUT
;*****
PRTI:
3096 F3        DI          ; DISABLE PROCESSOR INTERRUPTS
3097 3E0D      MVI     A,IENI ; GET ENABLE INPUT INTERRUPTS CONTROL WORD
3099 D3E7      OUT     CWR    ; OUTPUT TO CONTROL WORD REGISTER
309B C9        RET          ; RETURN TO CALLER
  
```



ISIS 8080 MACRO ASSEMBLER, V1.0  
OUTPUT DATA ROUTINE

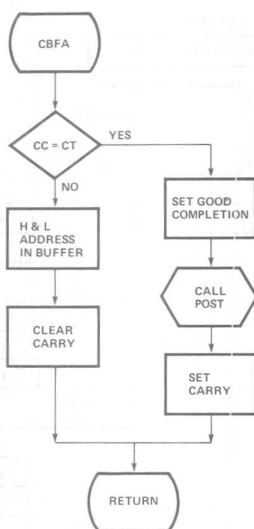
PAGE 8

```

;*****
;***** OUTPUT DATA ROUTINE
;*****
POUT:
309C DBE6      IN      PORTC ; GET PORTC STATUS
309E E620      ANI     IBFA  ; SEE IF OUTPUT BUFFER FULL
30A2 JNZ      PRTO  ; YES - BRANCH
30A3 CDBC30    CALL    CBFA  ; SET UP ADDRESS OF DATA
30A6 DAAF30    JC      PODOM  ; IF DONE - BRANCH
30A9 7E        MOV     A,M    ; GET DATA FROM BUFFER
30AA D3E4      OUT     PORTA ; OUTPUT DATA
30AC C39C30    JMP     POUT   ; LOOP

;*****
;***** END OF OUTPUT TRANSACTION
;*****
PODON:
30AF AF        XRA      A      ; CLEAR A REG
30B0 32EC30    STA     PRGWT+1 ; CLEAR WRITE IN PROGRESS
30B3 C38630    JMP     PRTO    ; RETURN

;*****
;***** RETURN FROM OUTPUT
;*****
PRTO:
30B6 F3        DI          ; DISABLE PROCESSOR INTERRUPTS
30B7 3E09      MVI     A,IENO ; GET ENABLE OUTPUT INTERRUPTS CONTROL WORD
30B9 D3E7      OUT     CWR    ; OUTPUT TO CONTROL WORD REGISTER
30BB C9        RET          ; RETURN TO CALLER
  
```



Setup Buffer Address Subroutine

ISIS 8080 MACRO ASSEMBLER, V1.0  
COMPUTE BUFFER ADDRESS ROUTINE

PAGE 9

```

;*****
;***** COMPUTE BUFFER ADDRESS ROUTINE
;*****
CBFA:
308C 210500 LXI H,CBCT ; GET INDEX TO CT
308F 19 DAD D ; COMPUTE ADDRESS OF CT
30C0 7E MOV A,M ; GET CT
30C1 34 INR M ; INC CT
30C2 28 DCR H ; DEC TO CC
30C3 BE CMP H ; SEE IF EQUAL
30C4 CAD530 JZ PCOMP ; IF EQUAL - DONE GO TELL USER
30C7 210200 LXI H,CBUP ; GET INDEX TO BUFFER ADDRESS
30CA 19 DAD D ; COMPUTE ADDRESS OF BUFFER ADDRESS
30CB 05 PUSH D ; SAVE D AND E REGISTERS
30CC 5E MOV E,M ; GET LSB OF BUFFER ADDRESS
30CD 23 INX H ; INC TO NEXT BYTE
30CE 56 MOV D,M ; GET BUFFER MSB
30CF AC XRA H ; CLEAR H REG
30D0 6F MOV L,A ; GET CT
30D1 19 DAD D ; COMPUTE CHARACTER ADDRESS
30D2 D1 POP D ; RESTORE CONTROL BLOCK ADDRESS
30D3 AF XRA A ; CLEAR CARRY
30D4 C9 RET ; RETURN TO CALLER
  
```

ISIS 8080 MACRO ASSEMBLER, V1.0  
POST TO USER COMPLETION ROUTINE

PAGE 10

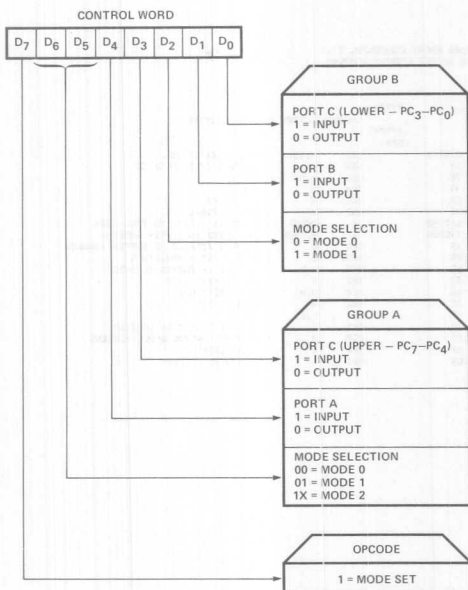
```

;*****
;***** POST GOOD COMPLETION TO USER
;*****
PCOMP:
30D5 3E00 MVI A,STOD ; GET GOOD STATUS CODE
30D7 CDDC30 CALL POST ; CALL USER ROUTINE
30DA 37 STC ; SET CARRY
30DB C9 RET ; RETURN TO CALLER

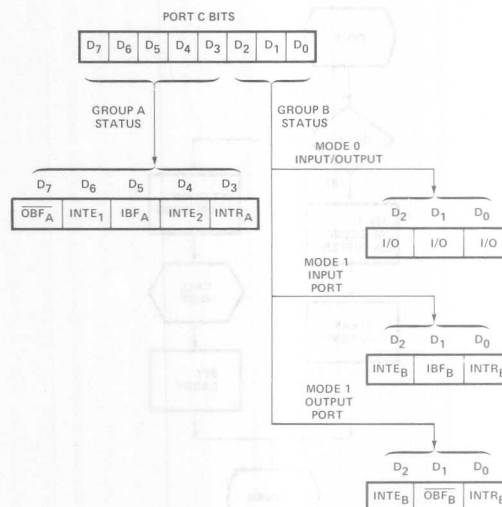
;*****
;***** POST TO USER COMPLETION ROUTINE
;*****
; INPUTS : STATUS CODE IN A REG
; CONTROL BLOCK ADDRESS IN D AND E REG
; OUTPUTS: PASSES CONTROL TO USER COMPLETION ADDRESS
; SPECIFIED IN CONTROL BLOCK
;*****
POST:
30DC EB XCHG
30DD 77 MOV M,A ; UPDATE STATUS
30DE EB XCHG
30DF 210600 LXI H,CBCT ; GET INDEX TO COMPLETION ADDRESS
30E2 19 DAD D ; COMPUTE ADDRESS
30E3 4E MOV C,M ; GET LSB OF COMPLETION ADDRESS
30E4 23 INX H ; INC TO NEXT BYTE
30E5 46 MOV B,M ; GET MSB BYTE OF COMPLETION ADDRESS
30E6 C5 PUSH B ; PUSH ADDRESS INTO STACK
30E7 C9 RET ; PASS CONTROL TO USER ROUTINE
30E8 C9 RET ; RETURN TO CALLER

;*****
;***** DATA AND TABLES
;*****
; IF DATA NON ZERO CONTROL BLOCK IN PROGRESS
;*****
30E9 0000 PRGDRD: DW 0 ; IN PROGRESS READ CONTROL BLOCK
30EB 0000 PRGWT: DW 0 ; IN PROGRESS WRITE CONTROL BLOCK
;*****
; END OF MASTER SOFTWARE DRIVER
;*****
0000 END
  
```

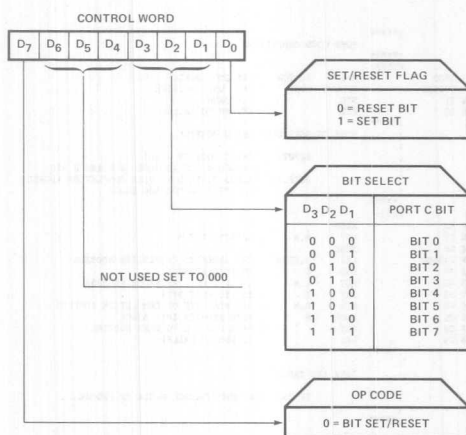
# APPENDIX A – 8255A QUICK REFERENCE



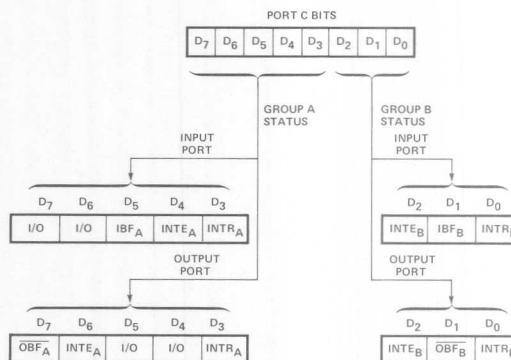
MODE CONTROL WORD



MODE 1 STATUS WORD

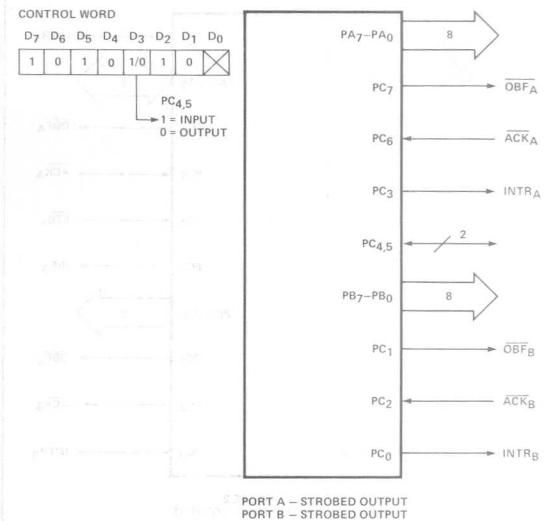
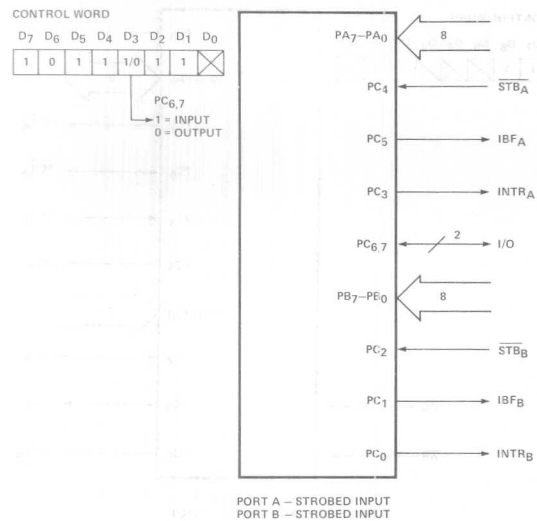
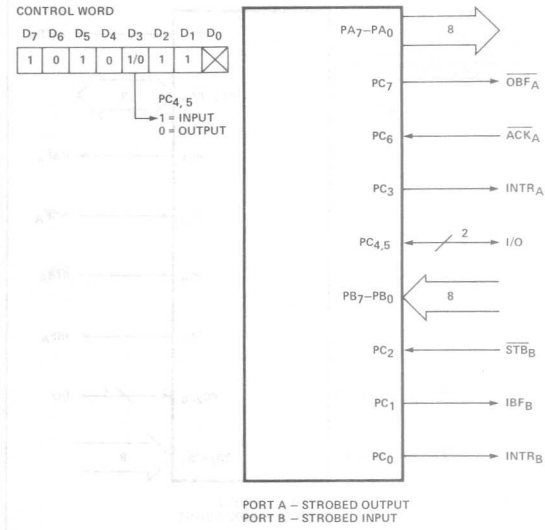
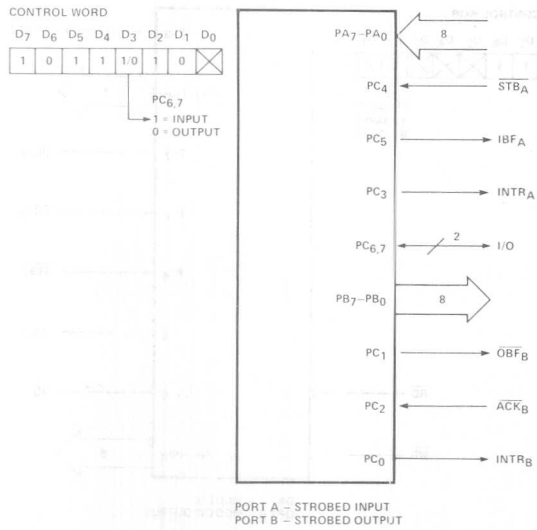


BIT SET/RESET CONTROL WORD

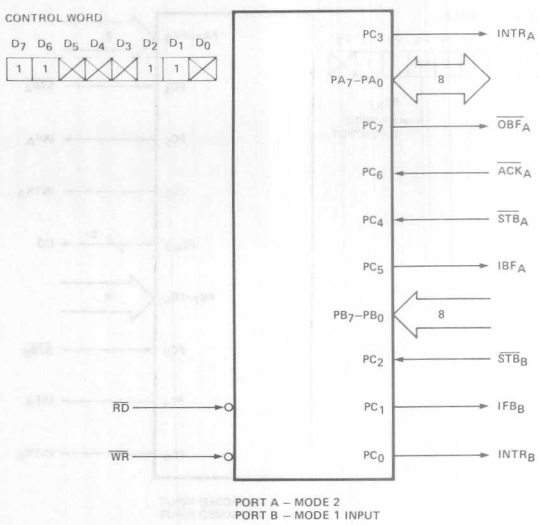
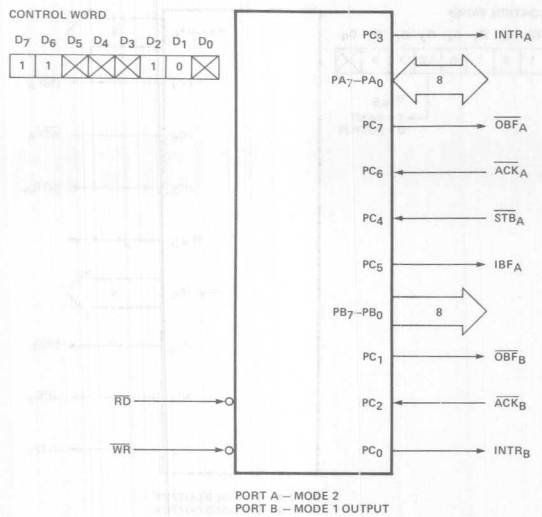
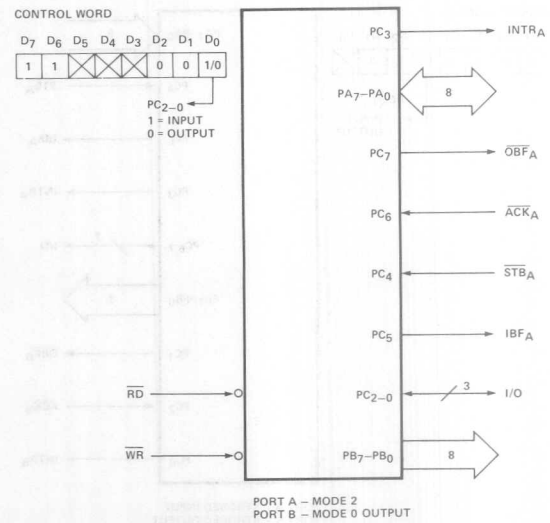
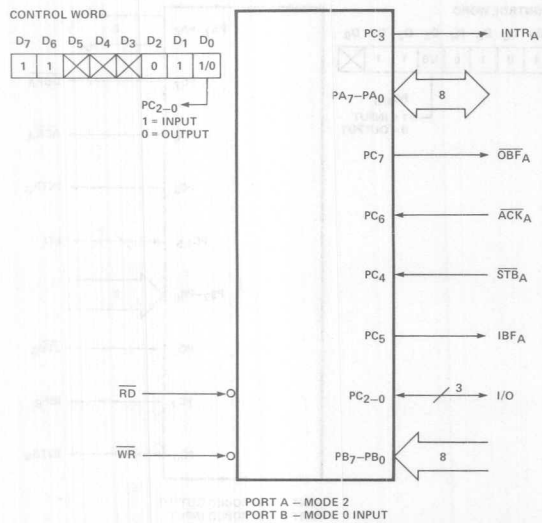


MODE 2 STATUS WORD

# MODE 1 CONFIGURATIONS



# MODE 2 CONFIGURATIONS



# Using The 8259 Programmable Interrupt Controller

by John Beaston

INTRODUCTION.....	2-94
CONCEPTS.....	2-94
8080 INTERRUPTS.....	2-95
8259—8080 OVERVIEW.....	2-96
8259 BLOCK DIAGRAM.....	2-97
INPUT CIRCUIT.....	2-98
PRIORITY CELL.....	2-99
DATA BUS BUFFER.....	2-100
READ/WRITE CONTROL LOGIC.....	2-100
CASCADE BUFFER/COMPARATOR.....	2-100
PIN DEFINITIONS.....	2-100
PROGRAMMING THE 8259.....	2-101
INITIALIZATION COMMAND WORDS (ICSWs).....	2-101
OPERATION COMMAND WORDS (OCWs).....	2-103
Fully Nested Mode.....	2-103
Rotating Priority Commands.....	2-104
Interrupt Masks (OCW1).....	2-106
Special Mask Mode (OCW3).....	2-106
Polled Mode (OCW3).....	2-107
Reading the 8259 Status (OCW3).....	2-107
CASCADING THE 8259.....	2-108
APPLICATION EXAMPLES.....	2-109
POWER FAIL/AUTO-START WITH BATTERY BACK-UP RAM.....	2-109
78 LEVEL INTERRUPT SYSTEM.....	2-114
CONCLUSION.....	2-118

## INTRODUCTION

The Intel® 8259 is a Programmable Interrupt Controller (PIC) designed for use in real-time, interrupt-driven microcomputer systems. The 8259 manages eight levels of interrupts and has built-in features allowing expandability up to 64 levels with the addition of other 8259s. A selection of programmable priority modes is available to reconfigure how the 8259 processes interrupt requests. Individual interrupt inputs may also be masked under software control. These modes and masks may be dynamically changed by the software at any time during program execution. This means that the complete interrupt structure can be defined as required, based on the total system environment. The 8259 is part of the MCS-80/85 Microcomputer Family and as such, it interfaces to the 8080/8085 system with a minimum of external hardware.

This application note explains the 8259 as a component and shows its use in two typical applications. These applications are an interrupt controlled power-fail/auto-start scheme for a microcomputer system with battery back-up RAM, and a >64 level interrupt-driven system. The battery back-up system will be described in detail and the conceptual software for the >64 level interrupt-driven system will be presented.

The first section of this application note introduces the concept of interrupts and reviews how interrupts are handled by the Intel® 8080A Microprocessor. It is fairly tutorial in nature, and may be skipped by the more knowledgeable reader. The second section describes the 8259 from a functional standpoint with explanation of the block diagram. Each device pin is explained in detail. The third section defines the various operating modes along with the specific software required. Short initialization and setup routines are given to illustrate the programming concepts. The fourth, and final, section describes the applications mentioned earlier.

## CONCEPTS

In microcomputer systems, there is usually a need for the processor to communicate with various Input/Output devices such as keyboards, displays, sensors, and other peripherals. From the system viewpoint, the processor should spend as little time as possible servicing the peripherals since the time required for these I/O chores directly affects the

amount of time available for other tasks. In other words, the system should be designed so that I/O servicing has little or no effect on the total system throughput. There are two basic methods of handling the I/O chores in a system: Status Polling and Interrupt Servicing.

The Status Poll method of I/O servicing essentially involves having the processor "ask" each peripheral if it needs servicing by testing the peripheral's status line. If the peripheral requires service, the processor branches to the appropriate service routine; if not, the processor continues with the main program. Clearly, there are several problems in implementing such an approach. First, how often a peripheral is polled is an important constraint. Some idea of the "frequency-of-service" required by each peripheral must be known and any software written for the system must accommodate this time dependence by "scheduling" when a device is polled. Second, there will obviously be times when a device is polled that is not ready for service, wasting the processor time that it took to do the poll. And other times, a ready device would have to wait until the processor "makes its rounds" before it could be serviced, slowing down the peripheral.

Other problems arise when certain peripherals are more important than others. The only way to implement the "priority" of devices is to poll the high priority devices more frequently than lower priority ones. It may even be necessary to poll the high priority devices while in a low priority device service routine. It is easy to see that the Polled approach can be inefficient both time-wise and software-wise. Overall, the Polled method of I/O servicing can have a detrimental effect on system throughput, thus limiting the tasks that could be performed by the processor.

A more desirable approach in most systems would allow the processor to be executing its main program and only stop to service the I/O when told to do so by the I/O itself. In effect, the device would asynchronously signal the processor when it required service. The processor would finish its current instruction and then jump to the service routine for the device requesting service. Once the service routine is complete, the processor would resume exactly where it left off in the main program.

This method of I/O servicing is called Interrupt. The status line of the peripheral is replaced by an

“interrupt request” line. Asserting this line signals the processor that service is needed. Using interrupts, no processor time is spent testing devices, scheduling is not needed, and priority schemes are readily implemented. It is easy to see that, using the Interrupt approach, system throughput would increase, allowing more tasks to be handled by the processor.

There are two basic methods of implementing the Interrupt approach: polled interrupts and vectored interrupts. Conceptually, in the polled interrupt method, the peripherals’ “interrupt request” lines are combinatorially OR’d into one line that interrupts the processor if any peripheral required service. The processor then polls each peripheral to determine the requesting device. In this scheme, the priority of the device is determined by its position in the polling sequence. Once the requesting device is found, the processor branches to the corresponding service routine. In contrast, vectored interrupts are those in which the requesting device supplies information which allows the processor to directly call the appropriate service routine. This method usually requires more hardware than the polled method. However, it allows much faster response to an interrupt since the polling time is eliminated. In simple vectored interrupt systems, all devices have the same priority. This is sometimes a limitation since the speed of the vectored method may be needed, while the prioritization of the polled method is also required; a flexible interrupt structure would have both.

In order to implement a truly flexible priority-vectored interrupt structure, a Programmable Interrupt Controller (PIC), such as the 8259, may be used. The 8259 functions as the overall manager of the interrupt-driven system and can implement both the polled and vectored interrupt structures. In the vectored structure it accepts interrupt requests from the peripherals, determines which of the incoming requests is the highest priority, ascertains whether the highest priority incoming request has higher priority than the interrupt level currently being serviced (if any) and then issues an interrupt to the processor based on the determination. Since each peripheral usually has a unique service routine associated with it, the PIC, after interrupting the processor, provides a “vectored” CALL instruction to point the processor directly to the service routine required by the interrupting device. In the polled structure, the same request priority determination is made, however software

polls the 8259 rather than the peripherals. When polled, the 8259 returns a data word indicating the highest priority peripheral requesting service. The software then uses this data word to branch to the appropriate service routine.

A variety of priority modes is a desirable feature of a PIC. Many options are conceivable; however, let’s describe a few which are available with the 8259 and will be mentioned later.

*Fully Nested* — Each input is assigned a priority. Interrupt Request input IR7 receives the lowest priority while IR0 receives the highest. A higher priority request will interrupt a lower priority service routine, but not vice versa. The lower priority service routine will be resumed upon completion of the higher priority routine. This is essentially a “general purpose” mode.

*Rotating Priority* — Like in the Fully Nested mode, each input is assigned a priority. However, when an interrupt occurs and the appropriate service routine is executed, the priorities are rotated so that the most recently serviced input has the lowest priority. Thus, if there are N inputs, a serviced peripheral will have to wait, in the worst case, until the other N-1 peripherals are serviced before receiving service again. This mode prevents “hogging” of the processor by a single peripheral and gives each input an equal chance at the processor.

*Specific Priority* — This mode is similar to the Rotating mode. The only difference is that the software can select the bottom priority input without an interrupt having to have occurred. Thus, the priority assignments may be changed at any time depending on the needs of the main program or the service routine.

In the 8259, these modes are programmable; that is, they may be changed dynamically under software control. Additionally, each mode may be modified by the use of interrupt masks. These masks allow individual inputs to be masked off; i.e., not be able to cause an interrupt regardless of its priority. Each mask is under software control.

Before we discuss how the 8259 handles interrupts, let’s digress slightly to review how the 8080 itself handles interrupt requests.

## 8080 INTERRUPTS

A peripheral device can initiate an interrupt to the 8080 by simply pulling the 8080’s Interrupt pin

fore an interrupt request may be asserted at any time. The 8080 can, however, enable and disable interrupts under software control by use of the Enable Interrupt (EI) and Disable Interrupt (DI) instructions. These instructions either set (EI) or reset (DI) an internal interrupt enable flip-flop. The output of this flip-flop is made available on the INTE (Interrupt Enabled) pin. Interrupts are disabled (INTE low) upon resetting the 8080.

At the end of each instruction cycle, the 8080 examines the state of the INT pin and the INTE flip-flop. If interrupts are enabled and an interrupt request is being made (both pins high), the 8080 enters an INTERRUPT machine cycle. During the INTERRUPT cycle, the 8080 resets the interrupt enable flip-flop (INTE goes low disabling response to further interrupts) and issues an Interrupt Acknowledge (INTA), by way of the System Controller 8228, to tell the interrupting device that it has the 8080's attention and may remove the INT assertion. In addition, the Program Counter (PC) is not incremented as it normally would be in normal machine cycles. This ensures that the 8080 can return to the pre-interrupt program location if the PC is saved. At this point, the 8080 expects the interrupting device to place an instruction on the data bus. The 8080 is, in effect, saying "Okay, now you have my attention. You are granted one wish. What will it be?" Any instruction may be used, but there are only two logical choices: a RESTART (RST) or a CALL. The reason one of these two should be used is that both put the program counter on the stack, allowing it to be restored after the interrupt service routine is complete.

When a CALL instruction is placed on the data bus in response to the Interrupt Acknowledge (INTA), the 8080 saves the program counter by pushing it onto the stack and then issues two additional INTAs by way of the 8228. In response, the interrupting device is expected to return two bytes which are the starting address of its service routine. The lower 8 bits of the address (LSB) are released at the first INTA and the higher 8-bits (MSB) are released at the second INTA. Execution then starts at this destination address. Using a CALL instruction in response to an interrupt is an extremely powerful tool in I/O servicing. However, a significant amount of hardware is usually required in order to ensure that the correct sequence of data is placed on the data bus. For systems not having a large number of peripherals, a special CALL in-

The RESTART (RST) instructions are actually special one-byte calls which have the destination address embedded within the 8-bit opcode. Executing an RST causes execution to be transferred (vectored) to one of eight fixed memory locations, see Figure 1. Any of these addresses may be used to store the first instructions of an interrupt service routine. In simple systems, the desired RST instruction can be generated by a simple 8-bit buffer external to the interrupting device. Since the RST instructions are calls, the old program counter contents are placed on the stack.

RST	HEX OP CODE	DESTINATION ADDRESS
RST 0	C7	00 H
RST 1	CF	08 H
RST 2	D7	10 H
RST 3	DF	18 H
RST 4	E7	20 H
RST 5	EF	28 H
RST 6	F7	30 H
RST 7	FF	38 H

Figure 1. RST Instruction Format

Return to the main program from an interrupt service routine is identical for both the CALL and the RST instructions. Assuming an equal number of pushes and pops from the stack during the service routine, the pre-interrupt program counter is on top of the stack at the end of the routine. Executing a RETURN (RET) instruction pops the top of the stack into the program counter, causing the main program to take up where it left off before receiving the interrupt. It is the service routine's responsibility to save and restore the processor registers and status as appropriate. Remember that interrupts are disabled after an Interrupt Acknowledge so an EI instruction must be executed in the service routine in order for the 8080 to respond to further interrupt requests.

## 8259-8080 OVERVIEW

Figure 2 shows the 8259-8080 system bus interface. It is recommended that an 8228 (or 8238)

System Controller and Bus Driver be used in conjunction with the 8080 when an 8259 is used to manage interrupts. This combination ensures that the 3 required  $\overline{INTA}$  pulses occur in response to an interrupt. Using the 8212 I/O Port as an 8080 status latch does not provide the necessary  $\overline{INTA}$  sequence.

The normal sequence of events that occur when an interrupt request is asserted is as follows:

1. One or more Interrupt Request lines (IRO—IR7) is raised high signaling the 8259 that peripheral service is being requested.
2. The 8259 accepts the requests, resolves the priorities, and sends an INT to the 8080.
3. The 8080 suspends the program flow at the end of the current instruction (INTE must be high), and issues an  $\overline{INTA}$  by way of the 8228.
4. Upon receiving the  $\overline{INTA}$ , the 8259 places a CALL instruction onto the data bus.
5. This CALL causes the 8080 to issue two additional  $\overline{INTA}$ s by way of the 8228.
6. These additional  $\overline{INTA}$ s allow the 8259 to release the address for the service routine of the interrupting peripheral onto the bus.
7. This completes the 3-byte CALL. Execution is vectored to the peripheral's service routine.

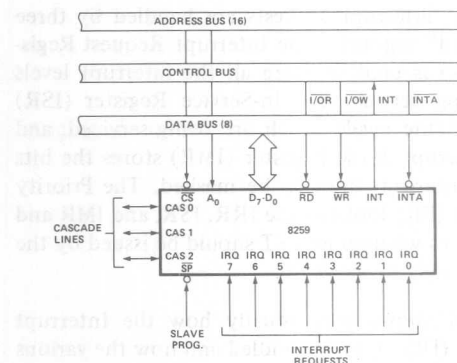


Figure 2. 8259 Interface to 8080 Standard System Bus

### 8259 BLOCK DIAGRAM

A block diagram of the 8259 is shown in Figure 3. As can be seen from the figure, the 8259 consists of eight major blocks: the Interrupt Request Register (IRR), the In-Service Register (ISR), the Interrupt Mask Register (IMR), the Priority Resolver (PR), the Cascade Buffer/Comparator, and logic blocks for Control and Read/Write. We'll go quickly over the individual blocks directly related to interrupt handling; the IRR, ISR, IMR, PR, and the Control logic. Then, by way of a conceptual diagram, we show how these various blocks interact. The remaining functional blocks are then discussed.

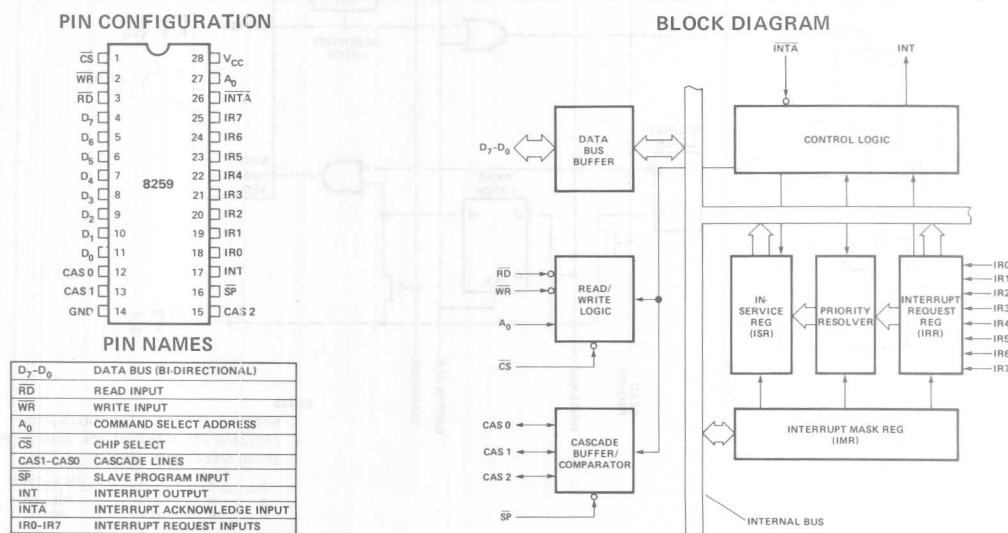


Figure 3. Block Diagram and Pin Configuration

Basically, interrupt requests are handled by three "cascaded" registers. The Interrupt Request Register (IRR) is used to store all the interrupt levels requesting service; the In-Service Register (ISR) stores all the levels which are being serviced; and the Interrupt Mask Register (IMR) stores the bits of the interrupt lines to be masked. The Priority Resolver (PR) looks at the IRR, ISR, and IMR and determines whether an INT should be issued by the Control logic to the 8080.

Figure 4 shows conceptually how the Interrupt Request (IR) input is handled and how the various registers interact. The figure represents one of eight "daisy-chained" priority cells; one for each IR input. The input circuitry is rather novel so it is discussed first.

### INPUT CIRCUIT

There are two classical ways of sensing an active interrupt request: a level sensitive or an edge sensitive input. A level sensitive input requires the request input go to the active state and remain active until that interrupt is acknowledged. This

structure is quite common and allows WIRE-OR'ed interrupt requests (the actual interrupting device must be determined via software as mentioned before). But (watch out!) the request must be removed shortly after acknowledgement or another, unwanted, interrupt could be generated.

The edge sensitive input requires only an inactive to active transition of the request input. This transition is saved in a flip-flop, so the active level need be maintained only long enough to serve as a clock pulse to the flip-flop. The level may remain active an arbitrarily long time without danger of generating an unwanted interrupt. It must ultimately return inactive before another active transition can be sensed. This structure is handy for handling interrupts from transient events, however it prevents WIRE-OR'ing since this connection does not provide the transitions needed. Be careful of edge inputs; noise on the request line could generate an erroneous interrupt.

The 8259 uses an edge lockout input which shares some characteristics with each of the above two techniques. The edge lockout input requires that a request transition from the inactive to the active

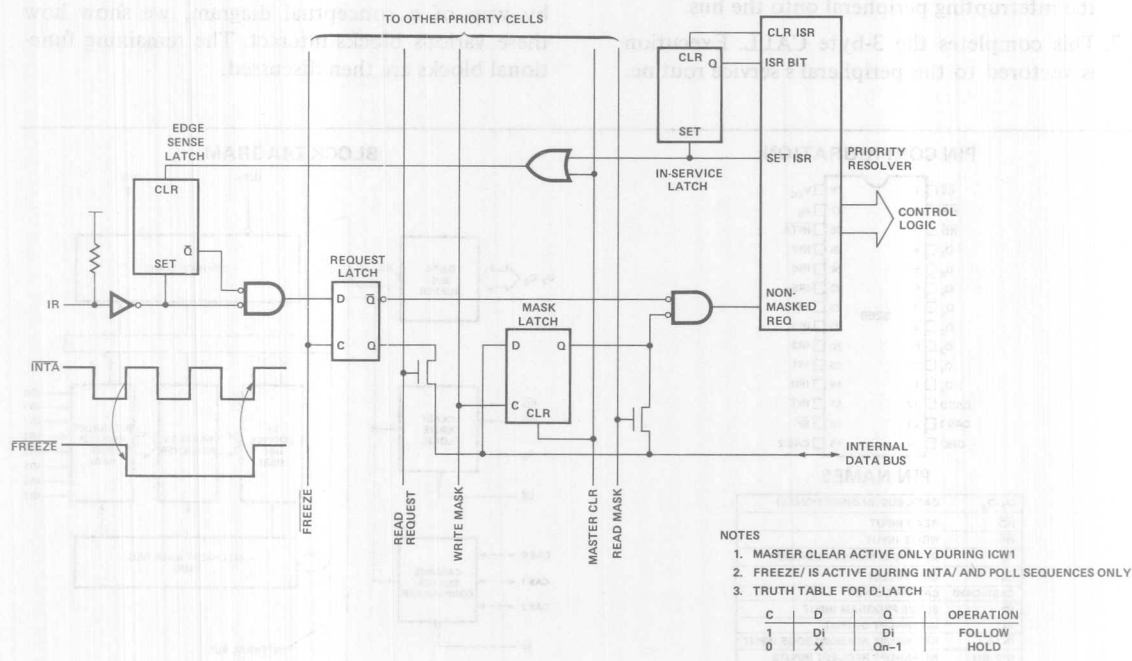


Figure 4. Priority Cell

state (as in edge sensitive) and then remain active (as in level sensitive) until the request is acknowledged. The inactive-to-active transition locks out all further requests on that input until the request has been acknowledged and the input has returned to the inactive state. Thus, the user need not worry about quickly removing the request after acknowledgement, in fear of generating a second interrupt. Figure 5 illustrates the timing required for the edge lockout input.

### PRIORITY CELL

Refer back to Figure 4 and follow an interrupt request thru the priority cell. First, notice that an inactive IR input sets the edge sense latch, arming that input. Then, an active IR input combinatorially propagates the request (assuming the input is not masked) to the Priority Resolver. The PR looks at the incoming requests and the currently in-service interrupts to ascertain whether an interrupt should be issued to the 8080. Assume for clarity that the request is the only one incoming and no requests are presently in service. The PR then causes the Control logic to pull the INT line to the 8080 high, interrupting the processor. When the 8080 is finished with the instruction being executed, it signals the 8228 to return an  $\overline{\text{INTA}}$ . This  $\overline{\text{INTA}}$  causes the 8259 to place a CALL instruction on the data bus and to freeze the IRR (note the  $\overline{\text{INTA}}$ -Freeze Request timing diagram). Thus, the

requesting IR input must remain active at least until after the first  $\overline{\text{INTA}}$ . With the input frozen and latched, the priority is again resolved by the PR, this time to determine the appropriate destination address for the CALL. The CALL instruction causes the 8080 to generate two additional  $\overline{\text{INTA}}$ s. During these  $\overline{\text{INTA}}$ s the destination address of the interrupt service routine is placed on the data bus by the 8259. (Don't worry for now about where the address comes from.) Immediately after the  $\overline{\text{INTA}}$  sequence, the PR then sets the corresponding bit in the ISR and simultaneously clears the edge sense latch, which clears the IRR bit. Notice the state of the edge sense latch (don't forget that the IR input may still be active). With the edge sense latch cleared, the still active IR input can not propagate thru the gate at the IRR input, thus further requests from this level are inhibited. The IR input must return to the inactive state, setting the edge sense latch and "opening" the IRR gate, before another request on the input can be recognized.

While off in the interrupt service routine, don't forget that the ISR bit is set. This prevents subsequent requests from this, and lower priority levels, from causing interrupts. It is the service routine's responsibility to clear the ISR bit with an End-of-Interrupt (EOI) command at the end of the service routine, telling the 8259 that it is complete. (How this is done is explained when 8259 programming is covered.)

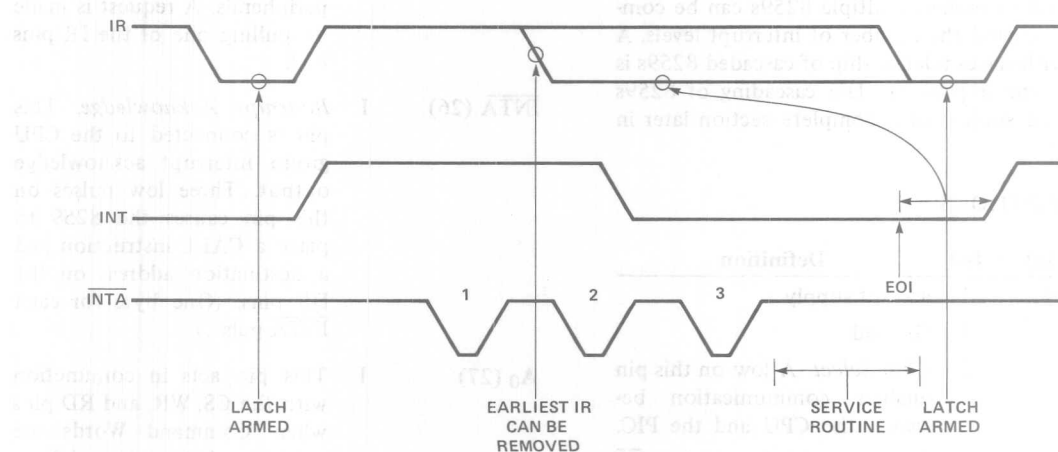


Figure 5. Edge Lockout Timing

masked; i.e., the Interrupt Mask Register bit was set? Nothing. The active state of the IR input would propagate thru the IRR but the set IMR bit would stop it before entering the PR. Thus, no interrupt could be generated. The IMR only acts on the output of the IRR, however, and if the program being executed somehow resets the IMR bit, the PR would then see our active request and an interrupt would be generated if appropriate.

Now that the functional blocks directly related to interrupt request processing have been discussed, let us discuss the remaining blocks.

## DATA BUS BUFFER

This 3-state, bidirectional, 8-bit buffer is used to interface the 8259 to the 8080 system data bus. Control words, status information, and the destination addresses are transferred through the Data Bus Buffer.

## READ/WRITE CONTROL LOGIC

The function of this block is to control the programming of the 8259 by accepting OUTPUT commands from the 8080. The Initialization and Operation Command Word Registers which store the various control formats are located in this block. Status reads are also controlled by this block using 8080 INPUT commands.

## CASCADE BUFFER/COMPARATOR

As alluded to earlier, multiple 8259s can be combined to expand the number of interrupt levels. A master-with-slaves relationship of cascaded 8259s is used for the expansion. The cascading of 8259s will be the subject of a complete section later in this note.

## PIN DEFINITIONS

Name (pin)	I/O	Definition
V <sub>CC</sub> (28)	I	+5 volt supply
GND (14)	I	Ground
$\overline{CS}$ (1)	I	<i>Chip Select.</i> A low on this pin enables communication between the CPU and the PIC.
$\overline{WR}$ (2)	I	A low on this input when $\overline{CS}$ is low enables the PIC to accept command words from the CPU.

RD (3)	I	A low on this input causes the PIC to output its status on the data bus when $\overline{CS}$ is low.
DB <sub>7</sub> -DB <sub>0</sub> (4-11)	I/O	The DB pins form a 3-state, bidirectional data bus which is connected to the CPU group (8080, 8224, 8228) data bus. Control and status information are transferred over this bus.
CAS <sub>0</sub> -CAS <sub>2</sub> (12,13,15)	I/O	<i>Cascade Lines.</i> The CAS pins form a private 8259 bus to control multiple 8259s. These pins are outputs for a master 8259 and are inputs for a slave 8259.
$\overline{SP}$ (16)	I	<i>Slave Program.</i> The state of this pin defines whether the 8259 is a master ( $\overline{SP}=1$ ) or a slave ( $\overline{SP}=0$ ). $\overline{SP}$ controls the I/O direction of the CAS pins.
INT (17)	O	<i>Interrupt.</i> This pin goes high whenever a valid interrupt request is asserted. INT is connected to the interrupt pin of the CPU.
IR <sub>0</sub> -IR <sub>7</sub> (18-25)	I	<i>Interrupt Request.</i> Interrupt requests are asserted by the peripherals. A request is made by pulling one of the IR pins high.
$\overline{INTA}$ (26)	I	<i>Interrupt Acknowledge.</i> This pin is connected to the CPU group interrupt acknowledge output. Three low pulses on this pin causes the 8259 to place a CALL instruction and a destination address on the DB pins. (One byte for each $\overline{INTA}$ pulse.)
A <sub>0</sub> (27)	I	This pin acts in conjunction with the CS, WR, and RD pins when Command Words are written and status is read from the 8259. It is typically connected to the CPU A <sub>0</sub> address line.

## PROGRAMMING THE 8259

As the name implies, the 8259 is programmable; operation is controlled via software thru command words. There are two types of command words used for the 8259: Initialization Command Words (ICWs) and Operation Command Words (OCWs).

### INITIALIZATION COMMAND WORDS (ICWs)

Before normal operation begins (i.e., after a system power-up), each 8259 in the system must be initialized by two or three ICWs. The ICWs tell each 8259:

1. If there are other 8259s in the system, and how they are connected.
2. The starting address of the service routines.
3. Whether the service routines are spaced 4 or 8 bytes apart.

Issuing an ICW1 starts the 8259 initialization sequence. Once started, the initialization sequence must be completed before the 8259 can process interrupt requests. This applies to each 8259 in a multiple 8259 system. During the initialization sequence, the following occur automatically:

1. Each edge sense circuit is reset. Thus an IR input must make an inactive to active transition, after initialization, to generate an interrupt.
2. The Interrupt Mask Register is reset (no IR inputs masked).
3. IR7 is assigned priority level 7.
4. The Status Read and Special Mask mode flip-flops (explained later) are reset.

Each IR input has an address in memory associated with it. It is this address that is placed on the bus by the 8259 in response to the INTA pulses after the CALL is placed on the data bus. The addresses for all eight IR inputs are formatted in equally spaced intervals of either 4 or 8 bytes. If the service routine for a device is short, it may be possible to fit the entire routine within an 8-byte interval. Usually, however, the service routines require more than 8 bytes and the 4-byte interval is used to store a Jump (JMP) instruction which directs the 8080 to the appropriate routine. The 8-byte interval maintains compatibility with current 8080 RESTART instructions software, while the 4-byte interval is best for a compact Jump table. For each 8259, the starting address for this 32 or 64-byte page is programmable during initialization and can be located

anywhere in the memory map, starting on an even page boundary. To form the 16 bits needed for each address, address bits A<sub>15</sub>–A<sub>6</sub> are user supplied in the ICWs and bits A<sub>4</sub>–A<sub>0</sub> are inserted by the 8259. A<sub>5</sub>'s generation depends upon whether 4 or 8-byte intervals are programmed. For 4-byte intervals, you program A<sub>5</sub> in ICW1. The 8259 supplies A<sub>5</sub> for the 8-byte interval selection. Figure 6 shows how the address is developed for each IR input.

REQUEST INPUT	4-BYTE INTERVAL— A15–A5 SUPPLIED IN ICW1 AND ICW2					8-BYTE INTERVAL— A15–A6 SUPPLIED IN ICW1 AND ICW2					
	A4	A3	A2	A1	A0	A5	A4	A3	A2	A1	A0
IR0	0	0	0	0	0	0	0	0	0	0	0
IR1	0	0	1	0	0	0	0	1	0	0	0
IR2	0	1	0	0	0	0	1	0	0	0	0
IR3	0	1	1	0	0	0	1	1	0	0	0
IR4	1	0	0	0	0	1	0	0	0	0	0
IR5	1	0	1	0	0	1	0	1	0	0	0
IR6	1	1	0	0	0	1	1	0	0	0	0
IR7	1	1	1	0	0	1	1	1	0	0	0

Figure 6. Address Development

The formats for ICW1 and ICW2 are shown in Figure 7. The 8259 interprets any command with A<sub>0</sub>=0, D<sub>0</sub>=0, and D<sub>4</sub>=1 as an ICW1. Note that address bit A<sub>0</sub> is used as an additional control input for all command words. Bits F and S are the only yet undefined bits. Bit F (Format) determines the CALL address interval. If F=1, then addresses are in 4-byte intervals; if F=0, then the interval is 8 bytes. Bit S (Single) indicates if there is more than one 8259 in the system. If S=1, there is only a single 8259; S=0 means multiple 8259s. ICW2 simply supplies the MSB of the address used as the start of the service routine page and is sent with A<sub>0</sub>=1.

If the system contains multiple 8259s (ICW1 bit S=0), an additional ICW is needed: ICW3. This word controls the master-slave relationship to ensure the correct 8259 places the service routine address on the bus. Multiple 8259 systems in general, and ICW3 in particular, are discussed in another section.

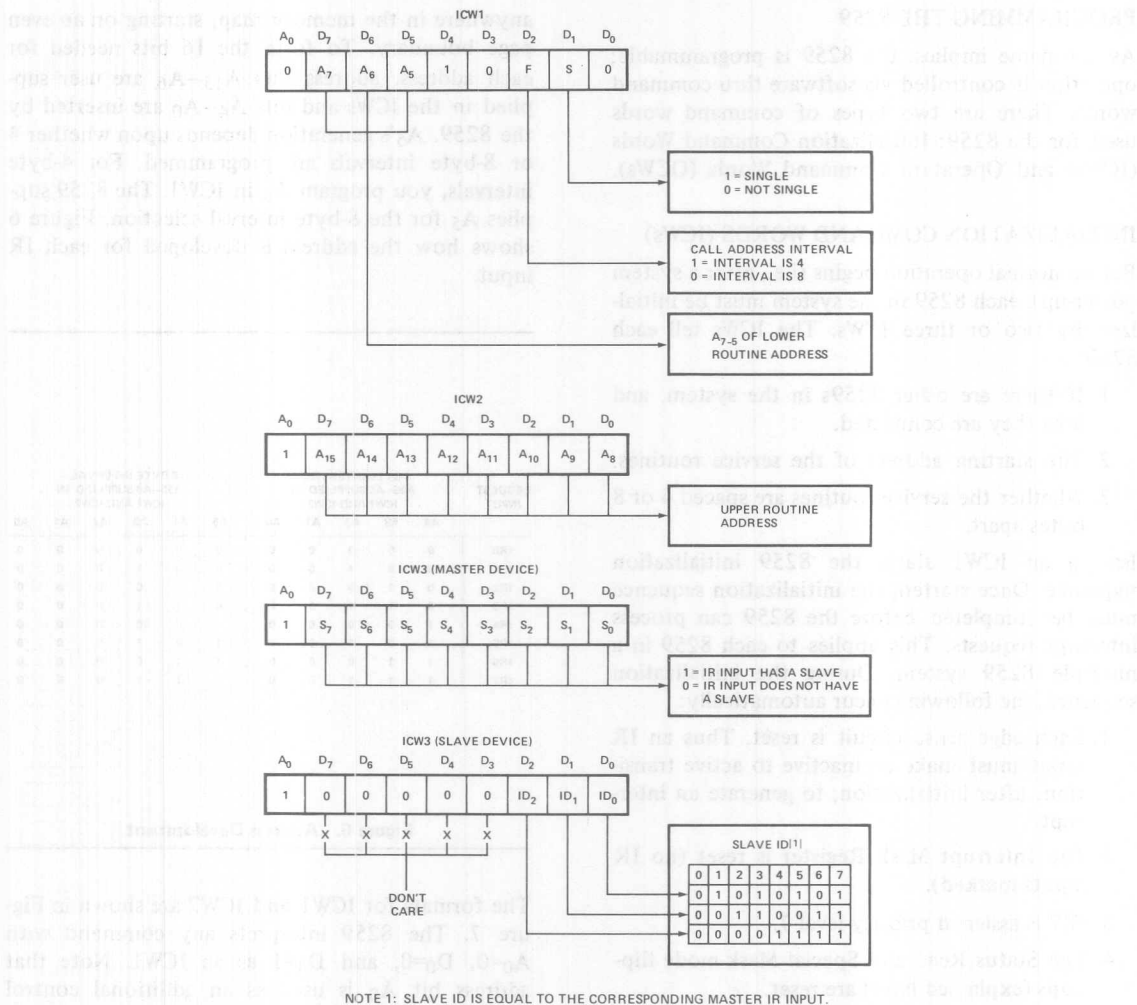


Figure 7. Initialization Command Word Format

Figure 8 shows the flow required for initialization. ICW1 is issued first, initiating the sequence. ICW2 must follow as the next command. With a single 8259, no ICW3 is required and the 8259 is ready to process interrupt requests immediately following ICW2. In order to ensure the integrity of any initialization or command sequence, interrupts must be disabled (by executing a DI instruction) over the initialization section of code. (Don't forget that interrupts are disabled automatically after the 8080 is reset.) Two typical initialization sequences are shown in Example 1.

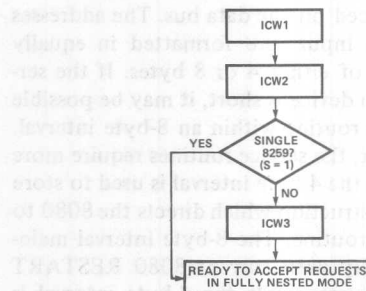


Figure 8. Initialization Flow

LOC	OBJ	SEQ	SOURCE STATEMENT
1		1	;
2		2	;
3		3	INITIALIZATION EXAMPLES
4		4	;
5		5	;
6	00DA	6	PT59A EQU 0DAH
7	00DB	7	PT59B EQU 0DBH
8		8	;
9		9	;
10		10	EXAMPLE: IN A SINGLE 8259 SYSTEM, THE 8259 IS INITIALIZED
11		11	FOR A 4-BYTE INTERVAL JUMP TABLE STARTING AT 3960H
12		12	;
13		13	;
14	0000 F3	14	INT591: DI ;DISABLE INTERRUPTS FOR COMMANDS
15	0001 3E76	15	MVI A,76H ;F=1,S=1, A6 & A5=1
16	0003 D3DA	16	OUT PT59A ;8259 PORT A6=0 ICW1
17	0005 3E39	17	MVI A,39H ;MSB CALL ADDRESS BYTE
18	0007 D3DB	18	OUT PT59B ;8259 PORT A6=1 ICW2
19	0009 FB	19	EI ;ENABLE INTERRUPTS
20		20	;
21		21	INITIALIZATION COMPLETE
22		22	;
23		23	;
24		24	EXAMPLE: WE WANT TO IMITATE THE RST INSTRUCTIONS
25		25	;
26		26	;
27	000A F3	27	INT592: DI ;DISABLE INTERRUPTS FOR COMMANDS
28	000B 3E02	28	MVI A,02H ;F=0,S=1, A7-A5=0
29	000D D3DA	29	OUT PT59A ;8259 PORT A6=0 ICW1
30	000F 3E00	30	MVI A,00H ;MSB CALL ADDRESS BYTE
31	0011 D3DB	31	OUT PT59B ;8259 PORT A6=1 ICW2
32	0013 FB	32	EI ;ENABLE INTERRUPTS
33		33	;
34		34	INITIALIZATION COMPLETE
35		35	;
36		36	;
37		37	;
38		38	;
39		39	END

### Example 1. Initialization Sequences

Once initialized, the 8259 is controlled using Operation Command Words. These words control the changing of priority modes, interrupt masks, and perform the End-of-Interrupt housekeeping.

### OPERATION COMMAND WORDS (OCWs)

After initialization, the 8259 is ready to accept interrupt requests on the IR inputs. However, during operation, the 8259 can be commanded to operate in a variety of priority modes through the Operation Command Words (OCWs). The various modes and their associated OCWs are described below.

### Fully Nested Mode

The 8259 handles requests in the Fully Nested mode without any OCW being written. In this mode, the IR inputs are assigned priorities such that IRO has the highest priority while IR7 has the lowest. When an interrupt is acknowledged, the highest priority request is determined and its address vector is placed on the data bus. In addition, the corresponding bit in the ISR is set. This bit remains set until an End-of-Interrupt command is received by the 8259 from the service routine. While the ISR bit is set, all further requests of the same and lower priority are inhibited from generating an interrupt to the 8080. Higher priority

requests can generate an interrupt. However, these interrupts are only acknowledged if the 8080 has enabled interrupts, by executing an EI instruction, since the preceding interrupt. Figure 9 illustrates this point.

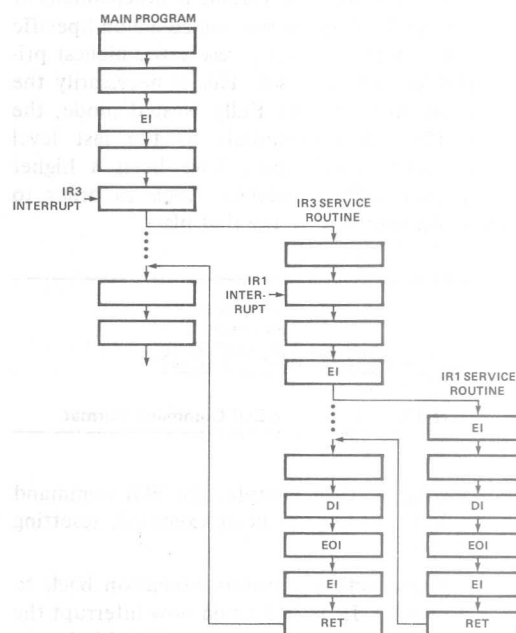


Figure 9. Fully Nested Example

During the main program, IR3 makes a request. Since interrupts are enabled, the 8080 is vectored to the IR3 service routine. During the IR3 routine, IR1 asserts a request. Since IR1 has higher priority than IR3, an interrupt is generated. Because the 8080 disabled interrupts in response to the IR3 interrupt, the IR1 interrupt is not acknowledged until an EI instruction is executed. Thus the IR3 routine has a "protected" section of code over which no interrupts are allowed. The IR1 routine has no such "protected" section since an EI instruction is the first one in its service routine.

What is happening to the ISR register? While in the main program, no ISR bits are set since no interrupts are in-service. When the IR3 interrupt is acknowledged, the ISR3 bit is set. When the IR1 interrupt is acknowledged, both the ISR1 and the ISR3 bits are set, indicating that neither routine is complete. At this time, only IRO could generate an

interrupt since it is the only higher priority input from those presently in-service.

To terminate the IR1 routine, the routine must inform the 8259 that it is complete by resetting its ISR bit. It does this by executing the EOI command. The format for this command is shown in Figure 10. Note that the format is independent of the interrupt level and is thus called a Non-Specific EOI. The command simply resets the highest priority ISR bit which is set. This is necessarily the correct bit since, in the Fully Nested mode, the highest ISR bit corresponds to the last level acknowledged; which must have been a higher priority than other in-service levels in order to generate the interrupt in the first place.

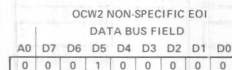


Figure 10. Non-specific EOI Command Format

Getting back to the example, the EOI command for the IR1 routine has been executed, resetting the ISR1 bit.

The RET instruction transfers execution back to the IR3 routine. IR0—IR2 could now interrupt the IR3 routine again, since only the IR3 bit in the ISR is set. No further interrupts occur in the example, so the Non-Specific EOI command in the routine resets the ISR3 bit this time and the RET instruction causes the main program to resume at the pre-interrupt location. One important thing to remember: the non-specific EOI command should only be used when in the Fully Nested mode. Other EOI-type commands are used when in other modes. Let us discuss those other modes now.

### Rotating Priority Commands

The Rotating Priority Commands serve in applications where the interrupting devices are of equal priority such as communication channels. The concept underlying rotating priority is that once a peripheral is serviced, all other equal priority peripherals should be given a chance to be serviced before the original peripheral is serviced again. This can be accomplished by assigning a peripheral the lowest priority after being serviced. Thus, in the worst case, the device would have to wait until all other devices are serviced before being serviced

again. OCW2 contains three commands which support rotating priority: two involve End-of-Interrupt [Rotate-at-EOI (Auto) and Rotate-at-EOI (Specific)] and one (Set-Priority), is independent of EOI. OCW2 contains one additional command which is not directly related to rotating priority but is sometimes used in conjunction with it: Specific EOI.

### Set-Priority Command

The Set-Priority Command in OCW2 allows the programmer to select the bottom priority device independently of an EOI; that is, without affecting the ISR. Figure 11 shows the format for the Set-Priority Command. L2, L1, and L0 code (in BCD) the IR input to be assigned the lowest priority. The priority of the remaining inputs are assigned accordingly. Example 2 illustrates the use of the Set-Priority Command.

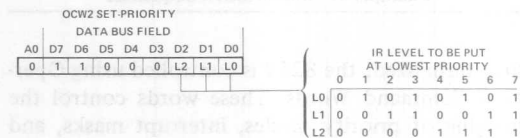


Figure 11. Set-Priority Command Format

EXAMPLE: STARTING WITH ANY PRIORITY STRUCTURE, ASSIGN IR2 PRIORITY LEVEL 4.  
BOTTOM PRIORITY IR6 CORRESPONDS TO IR2 BEING LEVEL 4, THUS L2 = 1, L1 = 1, AND L0=0 IN THE SET-PRIORITY COMMAND.

PRIORITY	BEFORE INPUT	AFTER INPUT
HIGHEST	5	7
	6	0
	7	1
	0	2
	1	3
	2	4
	3	5
LOWEST	4	6

### Example 2. Set-Priority Example

#### Rotate-at-EOI (Auto) Command

This command represents the "general purpose" implementation of Rotating Priority. When the Rotate-at-EOI (Auto) command is executed, the highest priority ISR bit is reset and priorities are rotated so that the request input of the ISR bit just reset is assigned the lowest priority. The format for the Rotate-at-EOI (Auto) command is shown in Figure 12. Since rotating priority implies that all peripherals are of equal importance, the service

routines are usually sacrosanct; that is, the EI instruction is placed at the end of the routine (after the EOI) to ensure that the routine will not be interrupted. Example 3 shows the effect of executing a Rotate-at-EOI (Auto) command.

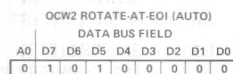
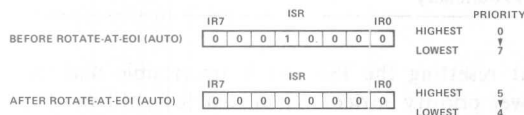


Figure 12. Rotate-at-EOI (Auto) Command Format

EXAMPLE: IR4 IS PRESENTLY IN SERVICE. WE WANT TO ROTATE IR4 TO BOTTOM PRIORITY AT EOI.



Example 3. Rotate-at-EOI (Auto)

When using the commands that rotate priorities, it is possible that the 8259 will not be able to determine the last level acknowledged (especially if nesting is allowed). If Rotate-at-EOI (Auto) is the only command used to reset ISR bits, then there is no problem. When a number of different commands are used a problem could occur. To prevent the 8259 from becoming confused, two commands that reset specific ISR bits are provided: the Rotate-at-EOI (Specific) and the Specific EOI commands.

### Rotate-at-EOI (Specific) Command

This command ensures that the correct ISR bit is reset at the end of a service routine because the bit to be reset is specified in the command itself. Additionally, the priorities are rotated so that the specified level is at the bottom. The format for the Rotate-at-EOI (Specific) command is shown in Figure 13. Example 4 illustrates this command.

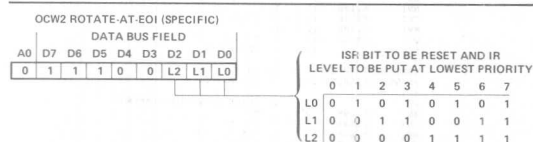
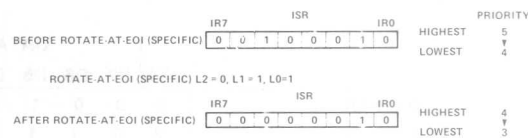


Figure 13. Rotate-at-EOI(Specific) Command Format

EXAMPLE: WE ARE IN THE IRS SERVICE ROUTINE AND WISH TO SET IR3 THE BOTTOM PRIORITY WHEN DONE.



Example 4. Rotate-at-EOI (Specific)

If the rotation of priorities is not desired, the Specific-EOI command is used.

### Specific-EOI Command

The Specific-EOI command is identical to the Rotate-at-EOI (Specific) command except that priorities are not rotated after the ISR bit is reset. The Specific-EOI command format is shown in Figure 14.

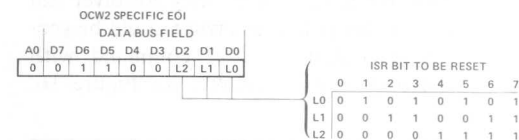


Figure 14. Specific EOI Command Format

In summarizing the various commands which reset ISR bits, some words of caution are appropriate. If only the Fully Nested mode is used, the Non-Specific EOI can be used without problems. For any other mode, it is good practice to use the End-of-Interrupt commands which specify the ISR bit to be reset. No additional code is required and the reassurance of an unconfused 8259 during system debug is worth the effort. The OCW2 command words are summarized in Figure 15.

## OCW2 COMMAND SUMMARY

COMMAND	DATA BUS FIELD									OPERATION
	A0	D7	D6	D5	D4	D3	D2	D1	D0	
NON-SPECIFIC EOI	0	0	0	1	0	0	0	0	0	RESET HIGHEST ISR BIT
SPECIFIC EOI	0	0	1	1	0	0	L2	L1	L0	RESET ISR SPECIFIED BY L2-L0
ROTATE-AT-EOI (AUTO)	0	1	0	1	0	0	0	0	0	RESET HIGHEST ISR BIT AND ASSIGN LOWEST PRIORITY
ROTATE-AT-EOI (SPECIFIC)	0	1	1	1	0	0	L2	L1	L0	RESET ISR SPECIFIED BY L2-L0 AND ASSIGN LOWEST PRIORITY
SET-PRIORITY	0	1	1	0	0	0	L2	L1	L0	SET L2-L0 LOWEST PRIORITY

Figure 15. OCW2 Command Summary

### Interrupt Masks (OCW1)

OCW1 controls the Interrupt Mask Register (IMR). Through OCW1, individual bits in the IMR may be set or reset by the software at any time. As stated earlier, the IMR acts only on the output of the Interrupt Request Register (IRR). Even with an IR input masked, it is still possible to set the IRR bit. However, no interrupt can be generated from the request since the IMR blocks the Priority Resolver from seeing the set IRR bit. If the IMR bit is reset while the IRR bit is set, the Priority Resolver can then see the IRR bit and an interrupt could be generated. After initialization, any command with A0=1 is interpreted as an OCW1, see Figure 16.

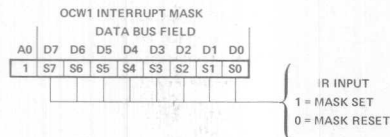


Figure 16. Interrupt Mask Command Format

### Special Mask Mode (OCW3)

The last Operation Command Word is OCW3. This word controls two additional modes plus the reading of the various registers. The first mode is the Special Mask Mode (SMM).

Let us say that you are in a service routine that contains a section of code where you want all interrupts enabled; that is, you want to allow your lower priority devices to generate interrupts. You could accomplish this by using an EOI command to reset the ISR bit corresponding to the routine we are in.

But resetting the ISR bit is irreversible and the lower priority devices remain enabled until another interrupt on your level occurred. The effect of the ISR bit can be temporarily suspended by first masking the input that is in-service and then setting the Special Mask Mode. Once SMM is set, it remains in affect until it is reset. The format to set and reset SMM is shown in Figure 17. The only requirements for SMM are that the level corresponding to the routine setting SMM must be masked through OCW1 and that interrupts are enabled. Example 5 shows how to enable interrupts over a particular section of code.

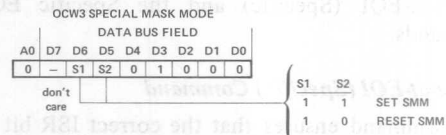


Figure 17. Special Mask Mode Command Formats

```

;EXAMPLE: IR4 IS IN-SERVICE AND WE WISH TO ENABLE LOWER
;PRIORITY INPUTS OVER A PARTICULAR SECTION OF CODE.
;
;IR4 SERVICE ROUTINE WHICH CONTAINS SPECIAL MASK MODE
;IR4:
EI          ;ENABLE INTERRUPTS
;
;1ST PART OF SERVICE ROUTINE -
;LOWER PRIORITY INPUTS DISABLED.
;
DI          ;DISABLE INTERRUPTS FOR COMMANES
MVI A,16H  ;MASK IR4
OUT PT59B  ;8259 PORT AB=1
MVI A,48H  ;SET SMM
OUT PT59A  ;8259 PORT AB=0
;
EI          ;ENABLE INTERRUPTS
;
;2ND PART OF SERVICE ROUTINE-
;LOWER PRIORITY INTERRUPTS ENABLED
;
DI          ;DISABLE INTERRUPTS FOR COMMANES
MVI A,68H  ;RESET SMM
OUT PT59A  ;8259 PORT AB=0
MVI A,08H  ;REMOVE MASK ON IR4
OUT PT59B  ;8259 PORT AB=1
EI          ;ENABLE INTERRUPTS
;
;3RD PART OF SERVICE ROUTINE -
;LOWER PRIORITY INPUTS DISABLED.
;MUST CONTAIN APPROPRIATE EOI.
;
RET          ;RETURN

```

Example 5. Special Mask Mode

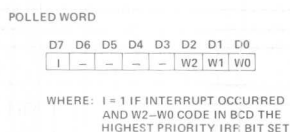
Note that SMM applies to all masked levels when set. If IR1 interrupts the IR4 routine in the above example while SMM is set, and then masks itself, IR2 and IR3 are enabled.

### Polled Mode (OCW3)

The 8259 also supports the polled interrupt method of I/O servicing mentioned earlier. Rather than having the processor poll the peripherals in order to find the actual interrupting device, the processor polls the 8259. This allows the use of all of the aforementioned priority modes. Additionally, both the polled and vectored interrupt methods can be used within the same program.

Basically, the polling is implemented by allowing the programmer to initiate a software controlled interrupt acknowledge through the "P" bit in OCW3. This interrupt acknowledge behaves exactly as the first "normal" hardware acknowledge; that is, the ISR bit of the highest priority input is set. The 8259 then enables a special word onto the data bus. This word shows whether an interrupt has occurred and what the highest IRR bit is.

To initiate a poll, interrupts must first be disabled; either by executing a DI instruction or from having an interrupt occur. Then an OCW3 with P=1 is sent to the 8259 using an OUTput command (or a WR pulse). The next  $\overline{RD}$  pulse (possibly from an INput command) is treated as an interrupt acknowledge, and the following word is placed on the data bus:



Service to the requesting device is achieved by the software decoding this word and branching to the appropriate service routine. Every time a poll is to be performed, the OCW3 must be written before the  $\overline{RD}$  pulse. If a poll is performed without an interrupt having occurred, the returned word is I=0 and W0, W1, and W2=1. The format for OCW3 Poll Command is shown in Figure 18.

To illustrate the Polled mode, consider a system where the 8259 and the 8080 are on different cards, and the system bus does not contain a line for the  $\overline{INTA}$  interrupt acknowledge, although interrupt request lines are provided. In this

instance, the Polled mode is the only way to take advantage of the 8259's prioritizing features. The INT pin of the 8259 is connected to the Interrupt Request line of the system bus while the 8259  $\overline{INTA}$  pin is simply held high. The 8080 card must contain logic to jam either a CALL or a RST instruction on the card's data bus in response to an interrupt on the system bus (either an 8259 on the processor card or an 8228 would accomplish this). The RST or the CALL vectors the 8080 to a polling routine. The polling routine simply writes an OCW3 with P=1 to the off-board 8259 port followed by an input at the same port. The 8259 then releases the above word onto the system data bus. The polling routine then decodes the returned word and vectors the 8080 to the appropriate service routine.

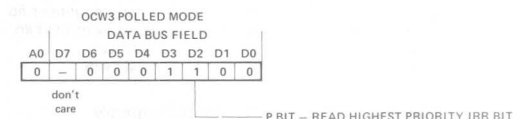


Figure 18. Polled Mode Command Format

This method can be extended to multiple off-board 8259s. Each 8259 is polled and the returned word indicates whether the selected 8259 is the one which generated the interrupt. Do not forget that even though the CALL features of the off-board 8259 are not being used, each 8259 must receive an initialization sequence. In this case, the starting address specified in the ICWs could be a "fake".

### Reading the 8259 Status (OCW3)

The contents of the IRR, the ISR, and the IMR can be read to update the user information on the system. The registers are read by issuing the appropriate OCW3 and then reading with an INput instruction or  $\overline{RD}$  pulse. The OCW3 words for reading the IRR and the ISR are shown in Figure 19.

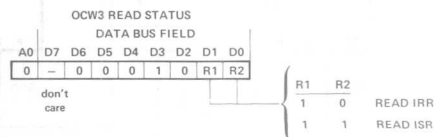


Figure 19. Read Status Command Formats

There is no need to write an OCW3 before every status read as long as the status read corresponds with the previous one; i.e., the 8259 "remembers" whether the ISR or the IRR has been previously selected by the OCW3.

For reading the IMR, an OUTput instruction (or  $\overline{WR}$  pulse) is not necessary to precede the INput instruction (or  $\overline{RD}$  pulse). The 8259 data lines contain the IMR whenever  $\overline{RD}$  is active and  $A_0=1$ . Thus an INput instruction to the 8259  $A_0=1$  port reads the IMR at any time.

A summary of OCW3 command words is shown in Figure 20.

COMMAND	OCW3 COMMAND SUMMARY									OPERATION
	A0	D7	D6	D5	D4	D3	D2	D1	D0	
POLL MODE	0	—	0	0	0	1	1	0	0	POLL ON NEXT $\overline{RD}$
READ ISR	0	—	0	0	0	1	0	1	1	READ ISR ON NEXT $\overline{RD}$
READ IRR	0	—	0	0	0	1	0	1	0	READ IRR ON NEXT $\overline{RD}$
SET SMM	0	—	1	1	0	1	0	0	0	SET SMM
RESET SMM	0	—	1	0	0	1	0	0	0	RESET SMM

Figure 20. OCW3 Command Summary

## CASCADING THE 8259

As mentioned earlier, more than one 8259 can be used to expand the priority interrupt scheme to up to 64 levels without additional hardware. In such cases, one 8259 acts as a master, and the others serve as slaves. Figure 21 shows a system contain-

ing a master and two slaves providing a total of 22 levels of interrupt.

Hardware-wise, the master is designated by a "high" on the  $\overline{SP}$  pin, while the  $\overline{SP}$  pins of the slaves are grounded. Additionally, the INT output pins of the slaves are connected to the IR input pins of the master. The CAS0–2 pins for all 8259s are paralleled. These pins act as outputs when the 8259 is a master and act as inputs for the slaves. The CAS0–2 pins serve as a private 8259 bus to control which slave has control of the system data bus when the destination address is issued to the 8080.

The sequence of events for a valid interrupt request on a slave is covered here. The slave IR input makes an inactive-to-active transition. Assuming this request is higher priority than other requests and in-service levels on the slave, the slave's INT pin is pulled high, signaling the master of the request. Assuming that this request to the master is higher priority than other master requests (possibly from other slaves) and master in-service levels, the master's INT pin is pulled high, interrupting the 8080. When this interrupt is acknowledged by the 8080, the master places the CALL instruction on the data bus. The master knows that the original request was on a slave (from ICW3 that will be covered shortly) and then puts the interrupted slave's ID on the CAS lines. This causes the slave to

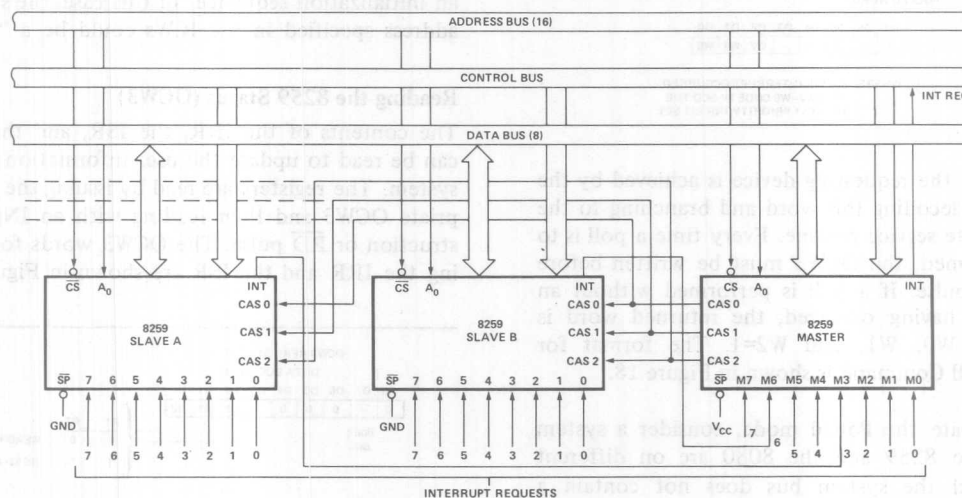


Figure 21. Cascaded System Diagram

place on the bus its preprogrammed address for the requesting input during the second and third INTAs. The appropriate ISR bits for both the master and slave are set. This completes the interrupt request.

Several things should become evident from the above sequence. First, because there are two ISR bits that are set by an acknowledged slave interrupt, two EOI commands must be issued; one for the master and one for the slave. And second, each 8259 must have a separate initialization sequence. This gives each IR input a unique address plus defines how the master and slaves are interconnected. This interconnection is specified in ICW3. The master ICW3 tells the master which of its IR inputs are connected to slaves. The slave ICW3 tells the slave which IR master input it is connected to. This IR input is the slave's ID. The format for ICW3 is shown in Figure 7. Also note that each slave could receive commands to operate in different modes; i.e., one slave could be in Rotating Priority while the other is in Fully Nested mode.

An initialization sequence is illustrated in Example 6. The master's jump table starts at 00H, slave A's at 20H, and slave B's at 40H; all with 4-byte intervals. The master ICW3 shows that there are slaves on IR inputs 3 and 6. Slave A ICW3 shows its ID as 3, indicating that it is the slave connected to the master IR3. Slave B's ID is 6 and it is connected to the master IR6. The priority levels are now arranged as shown.

		ICW									
		DATA BUS FIELD									
		A0	D7	D6	D5	D4	D3	D2	D1	D0	HEX
MASTER	ICW1	0	0	0	0	1	0	1	0	0	14
	ICW2	1	0	0	0	0	0	0	0	0	00
	ICW3	1	0	1	0	0	1	0	0	0	48
SLAVE A	ICW1	0	0	0	1	1	0	1	0	0	32
	ICW2	1	0	0	0	0	0	0	0	0	00
	ICW3	1	0	0	0	0	0	0	1	1	03
SLAVE B	ICW1	0	0	1	0	1	0	1	0	0	54
	ICW2	1	0	0	0	0	0	0	0	0	00
	ICW3	1	0	0	0	0	0	1	1	0	06

PRIORITY STRUCTURE						HIGHEST
LOWEST						
M7	SB7-SB0	M5	M4	SA7-SA0	M2	M1 M0

PRIORITY STRUCTURE

LOWEST                      HIGHEST

M7 SB7-SB0   M5   M4   SA7-SA0   M2   M1   M0

Example 6. Cascaded Initialization

Some special housekeeping software in the slave interrupt service routines is required in order to preserve a truly Fully Nested structure. Why? Notice that if level SA5 (IR5 on slave A) is in-service (both the Slave A ISR5 bit and the master

ISR3 bit are set) and level SA2 is asserted, then the priority structure of the slave will assert an interrupt to the master. But the master's ISR bit for that level is already set from the SA5 request. This will prohibit the request from being acknowledged until the master receives an EOI, thus losing the true Fully Nested structure since a request on SA2 should interrupt a SA5 service routine.

To solve this dilemma, the first task upon entering a service routine of a device connected to a slave is to mask off the lower priority master IR inputs. (in this case, M7, M6, M5, and M4). Then issue an EOI to the master for the input the slave is connected to (Specific EOI M3). This enables the master to accept higher priority interrupts from the slave. The masking process allows any interrupt request from a higher priority (higher than SA5) to be acknowledged and any lower priority request (M7 thru SA6) to be ignored. If the lower priority master inputs were not masked, the master would acknowledge a request on, for instance M7, since the M3 ISR bit is reset by the master EOI.

Software must also maintain the information that level SA5 is the lowest priority slave in-service. This is because the masks on the lower priority master inputs must be removed upon completing a service routine, but only by the lowest in-service slave level. If SA2 is the only in-service level then it resets the masks. However, in the main example, the SA2 routine returns to the SA5 routine. In this case, SA2 should not reset the masks, but allow SA5 to reset them just before returning. This can be accomplished by reading and saving the master IMR upon entering a slave input service routine and then restoring it upon leaving. Figure 22 is an example of how the SA5 service routine should look. This form should be followed for all service routines of devices connected to slave IR inputs.

## APPLICATION EXAMPLES

### POWER FAIL/AUTO-START WITH BATTERY BACKED-UP RAM

The first application illustrates the 8259 used in the Fully Nested mode in supporting a battery back-up scheme for the RAM (Random Access Memory) in a microcomputer system. Such a scheme is important in numerical and process control applications. The entire microcomputer system could be supported by a battery back-up scheme, however, due to the large amount of current usually required and the fact that most machinery is

the state of calculations and variables usually need to be saved. In the event of a loss of power, if these items are not already stored in RAM, they can be transferred there and saved using a simple battery back-up system.

LOC	OBJ	SEQ	SOURCE STATEMENT
1		1	
2		2	
3		3	EQUATES:
000B		4	NSPTB EQU 0DBH ;MASTER PORT WITH AB=1
000A		5	NSPTA EQU 0DAH ;MASTER PORT WITH AB=0
000A		6	SLPTA EQU 0EAH ;SLAVE PORT WITH AB=0
7		7	
8		8	
9		9	SAMPLE SAS SERVICE ROUTINE
000B D5		10	SAS: PUSH D ;SAVE D5
0001 C5		11	PUSH B ;SAVE B2
0002 E5		12	PUSH H ;SAVE H2
0003 P5		13	PUSH PSW ;SAVE A PLUS FLAGS
0004 080B		14	IN NSPTB ;READ MASTER INR
0006 5F		15	MOV E,A ;STORE IN E
0007 3E08		16	MVI A,0F0H ;MASK LOWER MASTER M7-M4
0009 0300		17	OUT NSPTB ;MASTER PORT WITH AB=1
000B 3E63		18	MVI A,63H ;SPECIFIC EOI TO M3
000D 03DA		19	OUT NSPTA ;MASTER PORT WITH AB=0
000F FB		20	EI ;ENABLE INTERRUPTS
21		21	
22		22	SAMPLE CAN NOW INTERRUPT ITSELF FOR HIGHER PRIORITY
23		23	INTERUPTS, ACTUAL SERVICE ROUTINE GOES HERE
24		24	
0010 F3		25	DI ;DISABLE INTERRUPTS FOR COMMANDS
0011 3E20		26	MVI A,20H ;NON-SPECIFIC EOI FOR SLAVE
0013 D3EA		27	OUT SLPTA ;SLAVE A PORT AB=0
0015 7B		28	MOV A,E ;RESTORE MASTER INR INTO A
0016 0300		29	OUT NSPTB ;MASTER PORT AB=1
0018 F1		30	POP PSW ;RESTORE A PLUS FLAGS
0019 E1		31	POP H ;RESTORE H2
001A C1		32	POP B ;RESTORE B2
001B D1		33	POP D ;RESTORE D5
001C FB		34	EI ;RE-ENABLE INTERRUPTS
001D C9		35	RET ;DONE, SO RETURN
36		36	
37		37	
38		38	END

Figure 22. Sample Slave Service Routine

The vehicle used in this application is the Intel® SBC 80/20 Single Board Computer. The SBC 80/20 contains an 8259 on-board along with control lines helpful in implementing the power-down and automatic restart sequence used in a battery back-up system. The SBC 80/20 also contains user-selectable jumpers which allow the on-board RAM to be powered by a supply separate from the supply used for the non-RAM components. Also, the output of an undedicated latch is available to be connected to the IR inputs of the 8259 (the latch is cleared via an output port). In addition, an undedicated, buffered, input line is provided, along with an input to the RAM decoder that will protect memory when asserted.

The additional circuitry to be described was constructed on an SBC 905 prototyping board. An SBC 635 Power Supply was used to power the non-RAM section of the 80/20 while an external DC

supplying power to the RAM. The SBC 635 was used since it provides an open collector ACLO output which indicates that the AC input line voltage is below 103/206 VAC (RMS).

The following is an example of a power-down and restart sequence that introduces the various power fail signals.

1. An AC power failure occurs and the ACLO goes high (ACLO is pulled up by the battery supply). This indicates that DC power will be reliable for at most 7.5 ms. The power fail circuitry generates a Power Fail Interrupt (PFI) signal. This signal sets the PFI latch, which is connected to the IRO input of the 8259, and sets the Power Fail Sense (PFS) latch. The state of this latch will indicate to the processor, upon reset, whether it is coming up from a power failure (warm start) or if it is coming up initially (cool start).
2. The processor is interrupted by the 8259 when the PFI latch is set. This pushes the pre-power-down program counter onto the stack and calls the service routine for the IRO input. The IRO service routine saves the processor status and any other needed variables. The routine should end with a HALT instruction to minimize bus transitions.
3. After a predetermined length of time (5 ms in this example) the power fail circuitry generates a Memory Protect (MPRO) signal. All processing for the power failure (including the interrupt response delays) must be completed within this 5 ms window. The MPRO signal ensures that spurious transitions on the system control bus caused by power going down do not alter the contents of the the RAM.
4. DC power goes down.
5. AC power returns. The power-on reset circuitry on the 80/20 generates a system RESET.
6. The processor reads the state of the  $\overline{\text{PFS}}$  line to determine the appropriate start-up sequence. The PFS latch is cleared, the MPRO signal is removed, and the PFI latch driving IRO is cleared by the Power Fail Sense Reset (PFSR) signal. The system then continues from the pre-power-down location for a warm start by restoring the processor status and

popping the pre-power-down program counter off the stack.

Figure 23 illustrates this timing.

Figure 24 shows the block diagram for the system. Notice that the RAM, the RAM decoder, and the power-down circuitry are powered by the battery supply.

The schematic of the power-down circuitry and the SBC 80/20 interface is shown in Figure 25. The design is very straightforward and uses CMOS logic to minimize the battery current requirements. The

Cold Start switch is necessary to ensure that during a cold start, the  $\overline{\text{PFS}}$  line is indicating "cold start" sense ( $\overline{\text{PFS}}$  high). Thus, for a cold start, the Cold Start switch is depressed during power on. After that, no further action is needed. Notice that the  $\overline{\text{PFI}}$  signal sets the on-board PFI latch. The output of this latch drives the 8259 IR0 input. This latch is cleared during the restart routine by executing an OUTput D4 H instruction. The state of the  $\overline{\text{PFS}}$  line may be read on the least significant data bus line (DB0) by executing an INput D4 H instruction. An 8255 Port (8255 #1, Port C, bit 0) is used to control the  $\overline{\text{PFSR}}$  line.

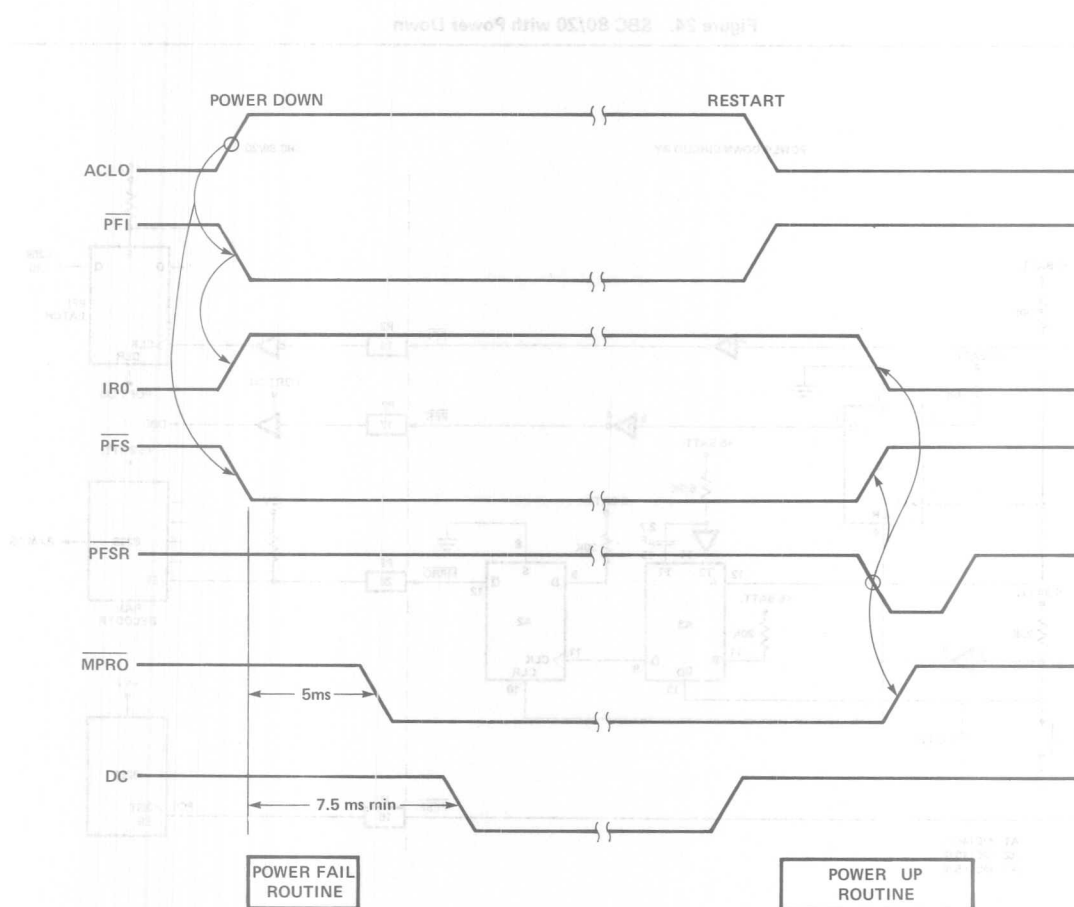


Figure 23. Power Down - Restart Timing

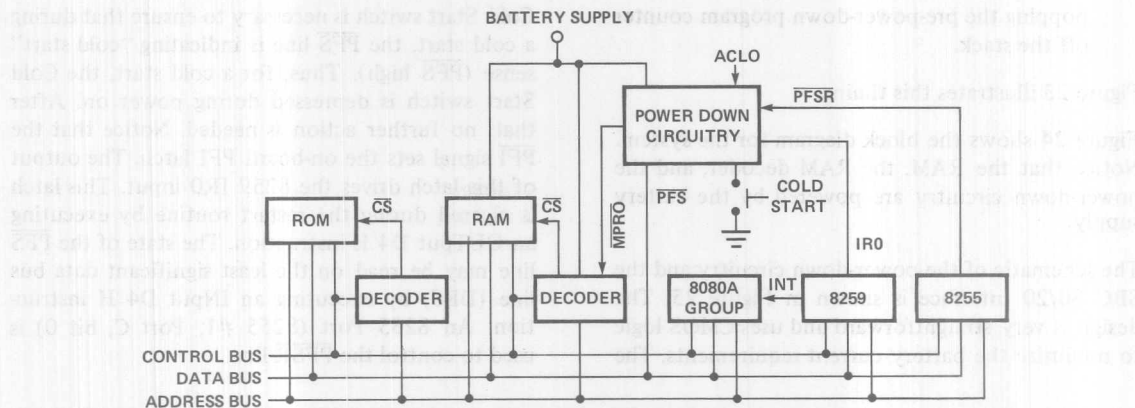


Figure 24. SBC 80/20 with Power Down

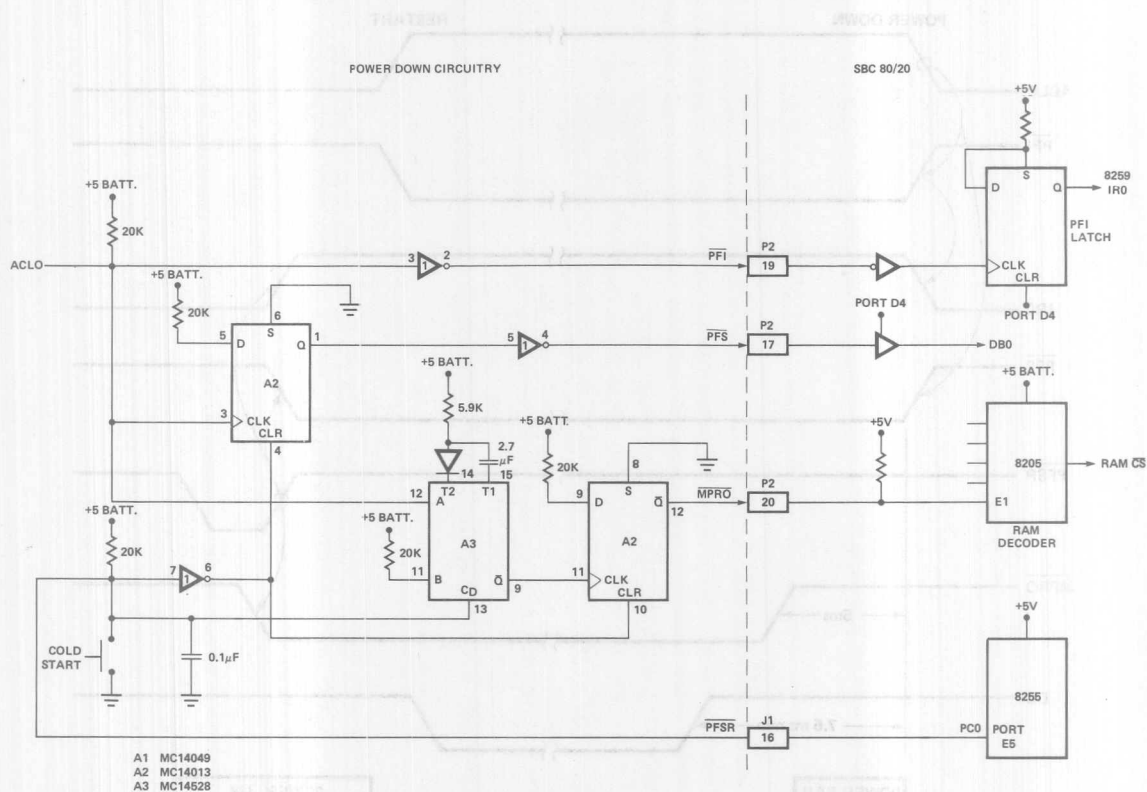


Figure 25. Power Down - SBC 80/20 Interface

The Fully Nested mode for the 8259 is used to ensure that IRO always has the highest priority. The remaining IR inputs can be used for any other purpose in the system. The only constraint is that the service routines must enable interrupts as early as possible. Obviously, this is to ensure that the power-down interrupt does not have to wait for service. If a rotating priority scheme is desired, another 8259 could be added as a slave and be pro-

grammed to operate in a rotating mode. The master would remain in the Fully Nested mode so that the IRO still remains the highest priority input.

The software to support the power-down circuitry is shown in Figure 26. The flow for each label will be discussed.

LOC	OBJ	SEQ	SOURCE STATEMENT
		1	;
		2	;POWER DOWN AND RESTART FOR THE SBC 88/28
		3	;
		4	;SYSTEM EQUATES:
00DA		5	PT59A EQU 0DAH ;8259 PORT WITH A8=0
00DB		6	PT59B EQU 0DBH ;8259 PORT WITH A8=1
00E7		7	PF1ICT EQU 0E7H ;8255 #1 CONTROL PORT
00E6		8	PF1IC EQU 0E6H ;8255 #1 PORT C
3000		9	SFSAVE EQU 3000H ;SP STORAGE IN RAM
0001		10	JPT EQU 01H ;MSB OF 8259 JUMP TABLE
		11	;
		12	;STARTING POINT AFTER SYSTEM RESET
		13	;
0008		14	ORG 08H
0008	0BD4	15	START: IN 0D4H ;READ PFS/ STATUS
0002	1F	16	PAR ;PFS/ ON DBE, PUT IN CARRY
0003	DA2001	17	JC CSTART ;PFS/1, THEN COLD START
		18	;
		19	;WSTART LOCATION. PFS/=0, THEN WARM START
		20	;
0006	3E80	21	WSTART: MVI A,08H ;SET 8255 #1 TO OUTPUT MODE
0000	D3E7	22	OUT PF1ICT ;8255 CONTROL PORT
		23	;
		24	;OUTPUT COMMAND MAKES PFS/ GO LOW WHICH REMOVES
		25	;MPRO/ AND CLEARS PFS LATCH
		26	;
000A	3001	27	MVI A,01H ;RETURN PFS/ HIGH
000C	D3E6	28	OUT PF1IC ;8255 #1 PORT C
000E	D3D4	29	OUT 0D4H ;RESET PFI LATCH
0010	CD1000	30	CALL INIT ;GO INITIALISE EVERYTHING
0011	2A0030	31	LHLD SFSAVE ;RETRIEVE SP FROM RAM
0016	F9	32	SPHL ;PUT BACK INTO SP
0017	C1	33	POP B ;RESTORE BC
0018	D1	34	POP D ;RESTORE DE
0019	E1	35	POP H ;RESTORE HL
001A	F1	36	POP PSW ;RESTORE A PLUS FLAGS
001B	F8	37	EI ;ENABLE INTERRUPTS
001C	C9	38	RET ;PRE-POWER-DOWN PC ON TOP
		39	;OF STACK SO RETURN TO IT
		40	;
		41	;INITIALIZATION ROUTINE. AT LEAST DO 8259....
		42	;
001D	3E16	43	INIT: MVI A,16H ;P=1,S=1,A7-A5=0 ICW1
001F	D3DA	44	OUT PT59A ;8259 PORT WITH A8=0
0021	3E91	45	MVI A,JPT ;MSB OF JUMP TABLE ICW2
0023	D3D0	46	OUT PT59B ;8259 PORT WITH A8=1
		47	;
		48	;ADD ANY OTHER INITIALIZATIONS HERE
		49	;
		50	RET ;RETURN
		51	;
		52	;POWER DOWN ROUTINE TO SAVE REGISTERS AND STATUS
		53	;
0026	F5	54	REGSAV: PUSH PSW ;SAVE A PLUS FLAGS
0027	E5	55	PUSH H ;SAVE HL
0028	D5	56	PUSH D ;SAVE DE
0029	C5	57	PUSH B ;SAVE BC
002A	210000	58	LXI H,0000H ;GET SET TO GET SP
002D	39	59	DAS SP ;SP NOW IN HL
002E	220030	60	SHLD SFSAVE ;SAVE SP IN RAM
		61	;
		62	;
		63	;EOI NOT REALLY NEEDED BUT INCLUDED FOR COMPLETENESS
		64	;
0031	3E20	65	MVI A,20H ;NON-SPECIFIC EOI
0033	D3DA	66	OUT PT59A ;8259 PORT WITH A8=0
0035	76	67	HLT ;HALT - GO DOWN GRACEFULLY
		68	;
		69	;8259 JUMP TABLE. ONLY IIRB IS USED, OTHERS DIRECTED TO RAM
		70	;
0100		71	ORG 100H
0100	C32600	72	JSTART: JMP REGSAV ;IIRB
0103	00	73	NOP ;3810H ;IIR1
0104	C31030	74	JMP 3810H ;IIR1
0107	00	75	NOP ;3820H ;IIR2
0108	C32030	76	JMP 3820H ;IIR2
0109	00	77	NOP ;3830H ;IIR3
010C	C33030	78	JMP 3830H ;IIR3
010F	00	79	NOP ;3840H ;IIR4
0110	C34030	80	JMP 3840H ;IIR4
0113	00	81	NOP ;3850H ;IIR5
0114	C35030	82	JMP 3850H ;IIR5
0117	00	83	NOP ;3860H ;IIR6
0118	C36030	84	JMP 3860H ;IIR6
011B	00	85	NOP ;3870H ;IIR7
011C	C37030	86	JMP 3870H ;IIR7
011F	00	87	NOP
		88	;
		89	;COLD START LOCATION. USER'S PROGRAM ENTERS HERE.
		90	;
0120	11003F	91	CSTART: LXI SP,3F80H ;INITIALIZE SP
0123	CD1000	92	CALL INIT ;INITIALISE EVERYTHING ELSE
0126	D3D4	93	OUT 0D4H ;RESET PFI LATCH
0128	F8	94	EI ;ENABLE INTERRUPTS
		95	;
		96	;USER PROGRAM STARTS HERE
		97	;
		98	END ;DONE

Figure 26. Power-down Software

After any system reset, the processor starts execution at location 0000H (START). The  $\overline{\text{PFS}}$  status is read and execution is transferred to CSTART if  $\overline{\text{PFS}}$  indicates a cold start (i.e., someone is depressing the Cold Start switch) or WSTART if a warm start is indicated ( $\overline{\text{PFS}}$  low). CSTART is the start of the user's program. The Stack Pointer (SP) and device initialization was included just to remind the reader that these must occur. The first EI instruction must appear after the 8259 has received its initialization sequence. The 8259 (and other devices) are initialized in the INIT subroutine. Four-byte intervals are selected for the 8259 since a jump table is being used ( $\text{F}=1$ ) and  $\text{S}=1$  since there is only one 8259 in the system. After initialization, the user's program is executed.

When a power failure occurs, execution is vectored by the 8259 to REGSAV by way of the jump table at JSTART. The pre-power-down program counter is placed on the stack. REGSAV saves the processor registers and flags in the usual manner by pushing them onto the stack. Other items, such as output port status, programmable peripheral states, etc., are pushed onto the stack at this time. The Stack Pointer (SP) could be pushed onto the stack by way of the register pair HL but the top of the stack can exist anywhere in memory and there is no way then of knowing where that is when in the power-up routine. Thus, the SP is saved at a dedicated location in RAM. It is not really necessary to include an EOI command in REGSAV since power will be removed from the 8259, but one is included for completeness. The final instruction before actually losing power is a HALT. This minimizes somewhat spurious transitions on the various busses and lets the processor die gracefully.

On reset, when a warm start is detected, execution is transferred to WSTART. WSTART activates  $\overline{\text{PFSR}}$  by way of the 8255 (all outputs go low when the 8255 is initialized). In the power-down circuitry,  $\overline{\text{PFSR}}$  clears the PFS latch and removes the  $\overline{\text{MPRO}}$  signal which then allows access to the RAM. WSTART also clears the PFI latch which arms the 8259 IRO input. Then the 8259 is re-initialized along with any other devices. The SP is retrieved from RAM and the processor registers and flags are restored by popping them off the stack. Interrupts are then enabled. Now the pre-power-down program counter is on top of the stack, so executing a RETurn instruction transfers the processor to exactly where it left off before the power failure.

Aside from illustrating the usefulness of the 8259 (and the SBC 80/20) in implementing a power failure protected microcomputer system, the above application should also point out a way of preserving the processor status when using interrupts.

## 78 LEVEL INTERRUPT SYSTEM

The second application illustrates the use of both the Fully Nested and Polled modes in implementing an interrupt structure with greater than 64 levels. The 8259 supports up to 64 levels with direct vectoring to the service routine. Extending the structure to greater than 64 levels requires the use of polling. A 78 level structure is used as an illustration, however the principles apply to systems with up to 512 levels.

To implement the 78 level structure, 3 tiers of 8259s are used. Nine 8259s are cascaded in the master-slave scheme giving 64 levels at tier 2. Two additional 8259s are connected, by way of the INT outputs, to two of the 64 inputs. The 16 inputs at tier 3, combined with the 62 remaining tier 2 inputs, give 78 total levels. The Fully Nested structure is preserved over all levels although direct vectoring is supplied for only the tier 2 inputs. Software is required to vector any tier 3 requests. Figure 27 shows the tiered structure used in this example. Notice that the tier 3 8259s are connected to the bottom level slave (SA7). This simplifies the housekeeping required in the service routines since the IMR of the master does not have to be changed as discussed in the cascading section. The master-slaves are interconnected as shown before, while the tier 3 8259s are connected as "masters"; that is, the SP pins are pulled high and the CAS pins are left unconnected. Since these 8259s are only going to be used in the polled mode, no  $\overline{\text{INTA}}$  is required, therefore the  $\overline{\text{INTA}}$  pins are pulled high.

The concept used to implement the 78 levels is to directly vector to all tier 2 input service routines. If a tier 2 input contains a tier 3 8259, the service routine for that input will poll the tier 3 8259 and branch to the tier 3 input service routine based on the word returned during the poll. Figure 28 shows how the jump table is organized assuming a starting location of 1000H and contiguous tables for all the tier 2 8259s. Note that "SA35" denotes the IR5 input of the slave connected to the master IR3 input. Also note that for the normal tier 2 inputs, the jump table vectors the processor directly to the service routine for that input, while for the

tier 2 inputs with 8259s, the processor is vectored to a service routine (i.e., SB0) which will poll to determine the actual tier 3 input requesting service. The polling routine utilizes the jump table starting at 1200H to vector the processor to the correct tier 3 service routine.

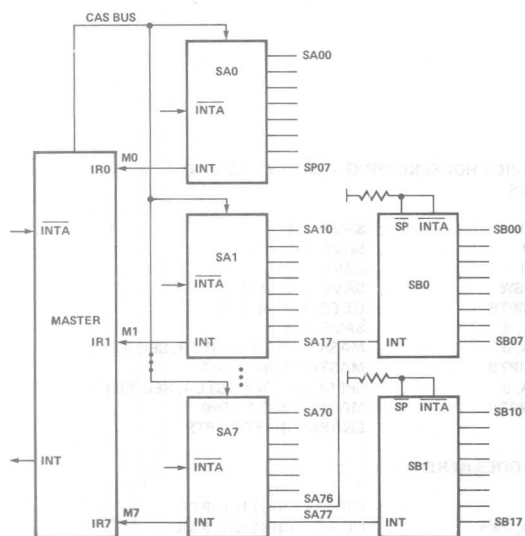


Figure 27. 78 Level Diagram

LOCATION	8259	CODE	COMMENTS
1000 H	SA0	JMP SA00	; SA00 SERVICE ROUTINE
101C H		JMP SA07	; SA07 SERVICE ROUTINE
1020 H	SA1	JMP SA10	; SA10 SERVICE ROUTINE
103C H		JMP SA17	; SA17 SERVICE ROUTINE
			; SA20-SA67 SERVICE ROUTINES
10E0 H	SA7	JMP SA70	; SA70 SERVICE ROUTINE
10F8 H		JMP SB0	; SB0 POLL ROUTINE
10FC H		JMP SB1	; SB1 POLL ROUTINE
1200 H	SB0	JMP SB00	; SB00 SERVICE ROUTINE
121C H		JMP SB07	; SB07 SERVICE ROUTINE
1220 H	SB1	JMP SB10	; SB10 SERVICE ROUTINE
123C H		JMP SB17	; SB17 SERVICE ROUTINE

Figure 28. Jump Table Organization

Each 8259 must receive an initialization sequence regardless of the mode. Since the tier 1 and 2 8259s are in cascade, they require all three ICWs. The tier 3 8259s require only ICW1 and ICW2 since only polling will be used on them and they

are connected as masters. The initialization sequence for each tier is shown in Figure 29. Notice that the master is initialized with a "dummy" jump table starting at 00H since all vectoring is done by the slaves. The tier 3 devices also receive "dummy" tables since only polling is used on tier 3.

```

; INITIALIZATION SEQUENCE
; INITIALIZE MASTER
MINT:  MVI  A, 14H      ; F-1, S-0, A7-A5=0 ICW1
        OUT  MPTA      ; MASTER PORT A0=0
        MVI  A, 00H    ; DUMMY ADR ICW2
        OUT  MPTB      ; MASTER PORT A0=1
        MVI  A, FFH    ; S7-S0=1 ICW3
        OUT  MPTB      ; MASTER PORT A0=1

; INITIALIZE SA SLAVES - X DENOTES SLAVE ID (SEE KEY)
SAXINT: MVI  A, 0      ; SEE KEY ICW1
        OUT  SAXPTA    ; SAXPORT A0=0
        MVI  A, 10H    ; ADR MSB ICW2
        OUT  SAXPTB    ; SAXPORT A0=1
        MVI  A, 0XH    ; SA ID ICW3
        OUT  SAXPTB    ; SAXPORT A0=1

; REPEAT ABOVE FOR EACH SA SLAVE
; INITIALIZE SB SLAVES - INITIALIZE AS MASTERS (SINGLE)
SB0INT: MVI  A, 16H    ; F=1, S=1, A7-A5=0 ICW1
        OUT  SB0PTA    ; SB0 PORT A0=0
        MVI  A, 00H    ; DUMMY ADR ICW2
        OUT  SB0PTB    ; SB0 PORT A0=1
SB1INT: MVI  A, 16H    ; F=1, S=1, A7-A5=0 ICW1
        OUT  SB1PTA    ; SB1 PORT A0=0
        MVI  A, 00H    ; DUMMY ADR ICW2
        OUT  SB1PTB    ; SB1 PORT A0=1

```

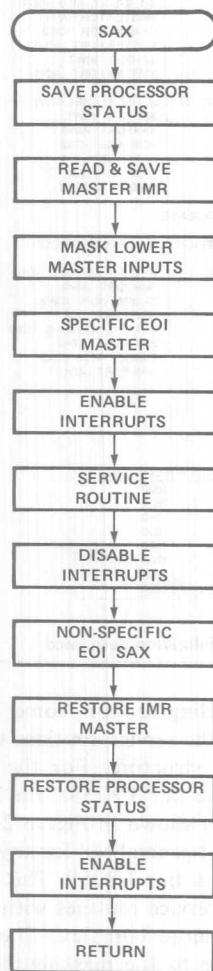
SAX	α (ICW1)	JUMP TABLE START (H)
0	14	1000
1	34	1020
2	54	1040
3	74	1060
4	94	1080
5	B4	10A0
6	D4	10C0
7	F4	10E0

Figure 29. Initialization Sequence

As shown in the cascading section, some house-keeping is required by the service routines to preserve the Fully Nested structure. For the tier 2 inputs which do not have tier 3 8259s, the house-keeping is similar to that shown in Figure 22. Figure 30 shows this format generalized for any tier 2 service routine without a tier 3 8259. The house-keeping for the tier 2 service routines with tier 3 8259s is only slightly more complex. The additional complexity is due to the masking required on the slave itself since the tier 2—tier 3 situation is analogous to the master-slave situation described in the cascading section. In this case, if for example, SB05 is in-service, the M7 and SA76 ISR bits must be reset and SA77 masked off to enable a higher priority input (SB04—SB00) to generate an interrupt. Figure 31 shows the form for the SA76 service routine (labeled SB0 in the jump table) which

service routine, by way of the jump table at 1200H, a separate return routine is used to end the

SB0RET restores the masks, executes an EOI, and restores the processor status.



GENERAL SAX SERVICE HOUSEKEEPING – USE KEY BELOW  
TO FIND VARIABLES

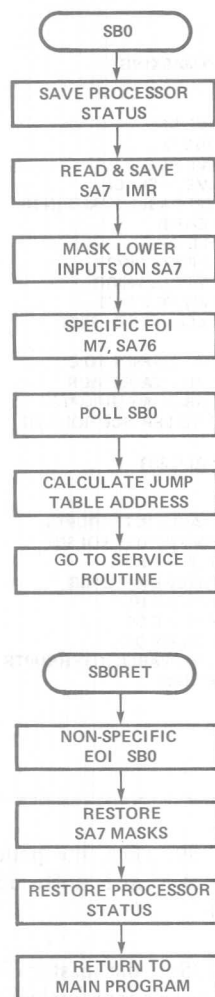
SAX :    PUSH D    ; SAVE DE  
         PUSH B    ; SAVE BC  
         PUSH H    ; SAVE HL  
         PUSH PSW   ; SAVE A+FLAGS  
         IN MPTB   ; GET MASTER IMR  
         MOV E, A   ; SAVE IN E  
         MVI A,  $\alpha$    ; MASK LOWER MASTER, SEE KEY  
         OUT MPTB   ; MASTER PORT A0=1  
         MVI A,  $\beta$    ; SPECIFIC EOI MASTER, SEE KEY  
         OUT MPTA   ; MASTER PORT A0=0  
         EI        ; ENABLE INTERRUPTS

SERVICE ROUTINE GOES HERE

DI        ; DISABLE INTERRUPTS  
MVI A, 20H   ; NON-SPECIFIC EOI SAX  
OUT SAXPTA   ; SAX PORT A0=0  
MOV A, E     ; RESTORE IMR MASTER  
OUT MPTB     ; MASTER PORT A0=1  
POP PSW      ; RESTORE A+FLAGS  
POP H        ; RESTORE HL  
POP B        ; RESTORE BC  
POP D        ; RESTORE DE  
EI        ; RE-ENABLE INTERRUPTS  
RET        ; DONE

KEY:	SAX	$\alpha$ (OCW1)	$\beta$ (OCW2)
	0	FE	60
	1	FC	61
	2	F8	62
	3	F0	63
	4	E0	64
	5	C0	65
	6	80	66
	7	00	67

Figure 30. Generalized Slave Service Routine



```

; SB0 ROUTINE -- FINDS REQUESTING INPUT AND
; BRANCHES TO CORRESPONDING SERVICE ROUTINE
;
SB0:  PUSH D           ; SAVE DE
      PUSH B           ; SAVE BC
      PUSH H           ; SAVE HL
      PUSH PSW         ; SAVE A+FLAGS
      IN  SA7PTB       ; READ SA7 IMR
      MOV D,A          ; SAVE IN D
      MVI A, 80 H      ; MASK SA77
      OUT  SA7PTB      ; SA7 PORT A0=1
      MVI A, 66H       ; SPECIFIC EO1 SA76
      OUT  SA7PTA      ; SA7 PORT A0=0
      MVI A, 67H       ; SPECIFIC EO1 M7
      OUT  MPTA        ; MASTER PORT A0=0
      LXI H, 1200H     ; JUMP TABLE START IN HL
      MVI B, 00H       ; CLEAR B
      MVI A, 0CH       ; POLL SB0
      OUT  SB0PTA      ; SB0 PORT A0=0
      IN   SB0PTA      ; GET POLL WORD
      ANI  07H         ; LIMIT TO 3 BITS
      ADD  A           ; GET TABLE OFFSET
      ADD  A           ;
      MOV  C,A         ; OFFSET IN C
      DAD  B           ; HL NOW HAS TABLE ADR
      PUSH D           ; SAVE IMR
      EI             ; ENABLE INTERRUPTS
      PCHL            ; JUMP TO ROUTINE VIA TABLE
;
; SB0RET -- DOES CLEAN UP AND EO1 AFTER SB0 INTERRUPT
;
SB0RET: DI           ; DISABLE INTERRUPTS
        MVI A, 20H   ; NON-SPECIFIC EO1 SB0
        OUT  SB0PTA  ; SB0 PORT A0=0
        POP  D       ; RESTORE IMR
        MOV  A,D     ; SA7 IMR
        OUT  SB0PTB  ; SB0 PORT A0=1
        POP  PSW     ; RESTORE PSW
        POP  H       ; RESTORE HL
        POP  B       ; RESTORE BC
        POP  D       ; RESTORE DE
        EI          ; RE-ENABLE INTERRUPTS
        RET         ; BACK TO MAIN PROGRAM
  
```

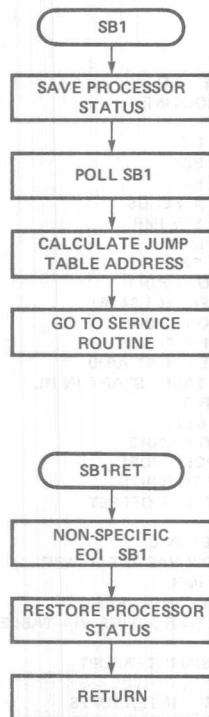
Figure 31. SB0 Housekeeping

The SB1 service routine can be simplified somewhat since it is the bottom priority and no masks need to be changed. Figure 32 shows the SB1 routine. Like the SB0 routine, a PCHL instruction is used to transfer execution, therefore a separate return routine is provided for all SB1 inputs.

The above format can be followed for any number of inputs up to the limit of 512 although once tier 3 8259s are connected to tier 2 8259s above the

master 7 input, it becomes necessary to include a section of code in the service routine to mask off and restore the master lower priority inputs.

This application has expanded the presentation of the cascading of 8259s and explained how to easily increase the number of interrupt levels by simply increasing the number of 8259s without adding additional hardware.



```

; SB1 ROUTINE -- LOWEST PRIORITY SO NO MASKING
; REQUIRED
SB1 :   PUSH D           ; SAVE DE
        PUSH B           ; AVE BC
        PUSH H           ; SAVE HL
        PUSH PSW          ; SAVE A+FLAGS
        LXI H,1220H       ; JUMP TABLE START IN HL
        MVI B,00H         ; CLEAR B
        MVI A,0CH         ; POLL SB1
        OUT SB1PTA        ; SB1 PORT A0=0
        IN SB1PTA         ; GET POLL WORD
        ANI 07H           ; LIMIT TO 3 BITS
        ADD A             ; CALCULATE OFFSET
        ADD A
        MOV C,A           ; MOVE OFFSET TO C
        DAD B             ; HL HAS TABLE ADR
        EI               ; ENABLE INTERRUPTS
        PCHL             ; GO TO SERVICE ROUTINE

; SB1RET -- DOES CLEAN UP AND EOI FOR ALL SB1
; INTERRUPTS
SB1RET: DI              ; DISABLE INTERRUPTS
        MVI A,20H        ; NON-SPECIFIC EOI SB1
        OUT SB1PTA       ; SB1 PORT A0=0
        POP PSW          ; RESTORE A+FLAGS
        POP H            ; RESTORE HL
        POP B            ; RESTORE BC
        POP D            ; RESTORE DE
        EI              ; RE-ENABLE INTERRUPTS
        RET              ; RETURN
  
```

Figure 32. SB1 Housekeeping

## CONCLUSION

This application note has explained the 8259 in detail and gives two applications illustrating the use of some of the numerous programmable features available. It should be evident from these discussions that the 8259 is an extremely flexible and easily programmed member of the Intel® MCS 80/85 Family.

## 8259 DESIGN HINTS

In Polled Mode, the CPU must write an OCW3 with the P bit set to inform the 8259 that it is being polled. The next RD actually performs the poll. It is truly the *next* RD; a RD to even another de-

vice will "poll" the 8259. Thus, the polled write and poll read must be adjacent in software for the poll to operate correctly.

When using cascaded 8259s, the master IRO input should not be used as a slave input unless *all* other master inputs (IR1–IR7) contain slaves. This restriction arises due to the CAS lines defaulting to 000 when valid requests are made on non-slave on IRO and cause it to drive the data bus with a CALL address in conflict with the master's CALL address. Of course, no conflict occurs if every master input contains a slave since each slave then has a unique CAS address.

# CRT Terminal Design Using The Intel® 8275 and 8279

by John Murray and George Alexy

---

1. INTRODUCTION .....	2-120
2. CRT SYSTEM DESIGN CONCEPTS .....	2-120
2.1 CRT OPERATION .....	2-120
2.2 MONITOR OPERATION .....	2-120
2.3 CRT TERMINAL DESCRIPTION .....	2-121
2.4 CRT TERMINAL IMPLEMENTATION .....	2-122
3. COMPONENT DESCRIPTION .....	2-123
3.1 8275 .....	2-123
3.2 8279 .....	2-128
4. CRT SYSTEM DESIGN EXAMPLE .....	2-129
4.1 SCOPE OF THE PROJECT .....	2-129
4.2 SYSTEM SPECIFICATIONS .....	2-130
4.3 SYSTEM HARDWARE DESIGN .....	2-130
4.3.1 General Considerations .....	2-130
4.3.2 Operation .....	2-131
4.3.3 System Timing .....	2-132
4.3.4 Dot Timing Logic .....	2-134
4.3.5 Keyboard Interface Design .....	2-136
4.3.6 System Memory Design .....	2-136
4.4 SYSTEM SOFTWARE DESIGN .....	2-137
4.4.1 General Considerations .....	2-137
4.4.2 Software Development .....	2-137
4.4.3 Operation .....	2-137
4.4.4 System Subroutines .....	2-141
APPENDIX 4.1 — CRT TERMINAL SCHEMATICS .....	
Serial Communications Section .....	2-150
CPU Section .....	2-151
Memory Section .....	2-153
Peripherals Section .....	2-155
Dot Timing Logic System .....	2-157
APPENDIX 5.2 — ESCAPE/CONTROL/DISPLAY CHARACTER SUMMARY .....	2-159
APPENDIX 5.3 — SUBROUTINE INTER-RELATIONSHIPS .....	2-160
APPENDIX 5.4 — SOFTWARE TIMING .....	2-161
APPENDIX 5.5 — VISUAL ATTRIBUTE IMPLEMENTATION CONSIDERATIONS .....	2-161
APPENDIX 5.6 — SOFTWARE LISTINGS .....	2-164

---

the reader with the conceptual and factual tools needed to apply the 8275 Programmable CRT Controller and 8279 Programmable Keyboard/Display Interface in CRT system design. The 8275 Controller is designed to interface CRT raster scan displays with Intel® Microcomputer Products. Its primary functions include refreshing the CRT display by buffering information from display memory and generating horizontal and vertical timing signals used for CRT synchronization. The programmable features of the 8275 allow it to be interfaced to almost any raster scan display with a minimum of external hardware. In addition, visual attribute features allow the implementation of specialized graphic display functions and display enhancement operations. The 8279 Keyboard Interface provides key scanning, debounce, and buffering features required for interfacing CRT terminal keyboards to the system processor. Two key or N-key rollover is provided. The use of these devices in a microcomputer based CRT terminal yields substantial savings in component count, printed circuit board area, and power consumption.

The application note is divided into five sections:

1. Introduction
2. CRT System Design Concepts
3. Component Description
4. CRT System Design Example
5. Appendix

Readers desiring an overview of CRT system design should consider reading the first three sections of the application note. Individuals requiring an in-depth knowledge of CRT system design should read the first three sections, then proceed to the design example. The design example consists of a description of the design of a complete CRT terminal. Both hardware and software aspects of the design are included. It will be assumed in Section 4 that the reader is familiar with the 8275, 8279, and 8257 data sheets, and the operation of the 8080A microprocessor.

## 2. CRT SYSTEM DESIGN CONCEPTS

### 2.1 CRT OPERATION

In order to fully understand the CRT terminal design process, it is necessary to consider the fundamentals of CRT operation. A typical CRT Monitor is shown in Figure 2-1. The CRT consists of an

frontal region (screen). A filament contained in the narrow cylindrical region (neck) of the CRT heats the cathode, causing the cathode to give off electrons by thermionic emission. Heating is accomplished by applying a low voltage source across the filament leads. A high voltage source applied between the cathode and the screen electrode (anode) accelerates the electrons toward the screen. The electron beam, upon striking the phosphorescent inner surface of the screen, produces light. To control the point at which the beam strikes the screen, two primary deflection techniques are utilized. The first technique, electromagnetic deflection, involves applying a current through a deflection coil placed around the neck of the CRT. The resulting magnetic field forces the electron beam to be deflected in proportion to the magnitude of the applied current. Electrostatic deflection involves placing deflection electrodes in the neck of the CRT perpendicular to the electron beam. An applied voltage changes the position of the beam accordingly.

### 2.2 MONITOR OPERATION

A CRT monitor consists of a CRT and the electronics required for positioning the beam in the desired manner. A block diagram of the control electronics contained within a typical CRT monitor is provided in Figure 2-2.

The horizontal oscillator is designed to move the electron beam horizontally across the CRT screen and then return the beam rapidly to its original position. As the beam is moved horizontally, the vertical oscillator causes the beam to be deflected vertically. The net result of these operations is to move the beam in a manner shown in Figure 2-3. If the intensity of the electron beam is modulated in a controlled manner as the beam sweeps across the screen, it is possible to display pictorial information on the CRT screen surface. It will be assumed that the monitor in question will be used for displaying alphanumeric characters or graphic symbols. In this case, the electron beam will be turned on to display a light region on the screen and turned off to display a dark region. Display information appearing at the video input to the CRT is applied through the video amplifier to a control grid located in the neck of the CRT. The magnitude of the video signal determines whether the electron beam will be on or off.

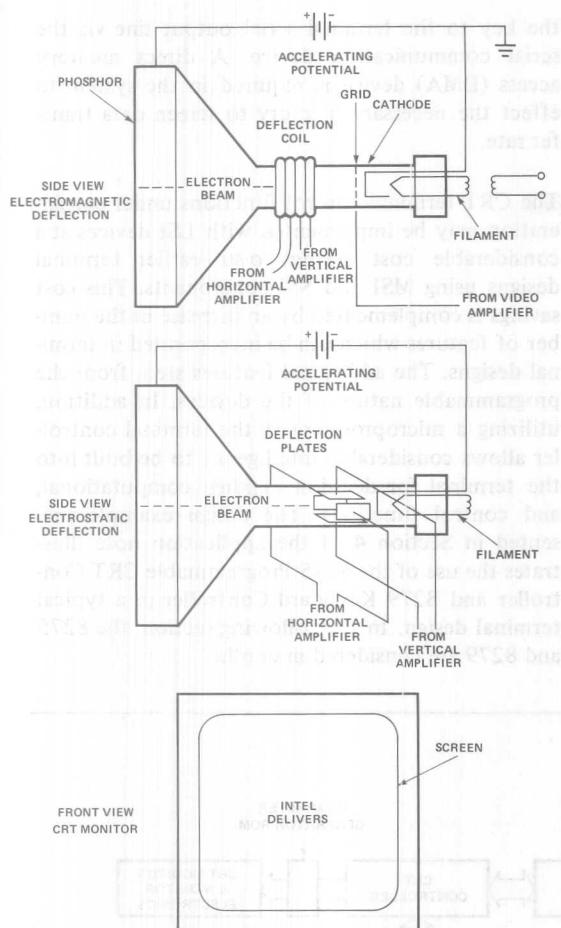


Figure 2-1. CRT Monitor

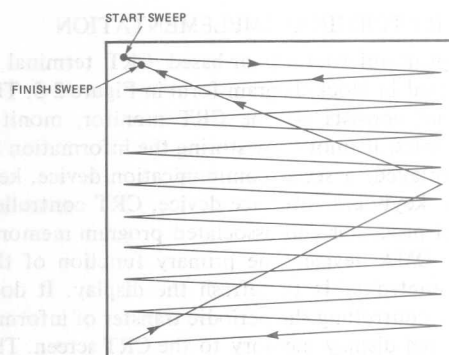


Figure 2-3. CRT Monitor Raster

### 2.3 CRT TERMINAL DESCRIPTION

A CRT terminal consists basically of a CRT monitor, monitor control electronics, memory for storing display information, logic to control information transfer to and from external devices and between internal devices, and a keyboard. The fundamental operations performed by a CRT terminal consist of the display of information contained in internal memory on the CRT screen, communication with manual data entry devices such as keyboards or light pens, and communication with external intelligent devices such as computers or data communication terminals. Typical CRT terminal communication functions are illustrated in Figure 2-4.

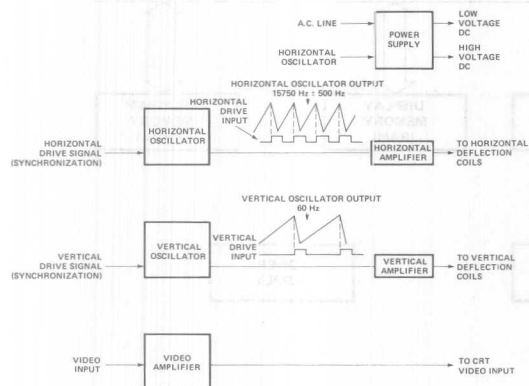


Figure 2-2. CRT Monitor Electronics

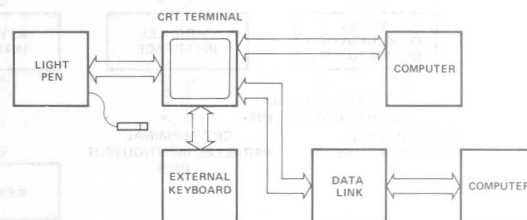


Figure 2-4. CRT Terminal Communications

## 2.4 CRT TERMINAL IMPLEMENTATION

A typical microprocessor-based CRT terminal is presented in block diagram form in Figure 2-5. The terminal consists of the CRT monitor, monitor electronics, memory for storing the information to be displayed, a serial communication device, keyboard, keyboard interface device, CRT controller, central processor and associated program memory, and a DMA device. The primary function of the CRT controller is to refresh the display. It does this by controlling the periodic transfer of information from display memory to the CRT screen. The central processor unit (CPU) coordinates the transfer of information to and from the terminal peripheral devices and external devices. When information from an external device is received by the terminal, the central processor performs character recognition and handling functions, display memory management functions, and cursor control functions. The CPU also interrogates the keyboard interface device. If a key depression is detected by the keyboard interface device, the CPU responds by transmitting the ASCII character representing

the key to the terminal serial output line via the serial communication device. A direct memory access (DMA) device is required in the system to effect the necessary memory to screen data transfer rate.

The CRT terminal control functions under consideration may be implemented with LSI devices at a considerable cost savings over earlier terminal designs using MSI and SSI components. This cost savings is complemented by an increase in the number of features which can be incorporated in terminal designs. The additional features stem from the programmable nature of the devices. In addition, utilizing a microprocessor as the terminal controller allows considerable intelligence to be built into the terminal for decision making, computational, and control functions. The design example presented in Section 4 of the application note illustrates the use of the 8275 Programmable CRT Controller and 8279 Keyboard Controller in a typical terminal design. In the following section, the 8275 and 8279 are considered in depth.

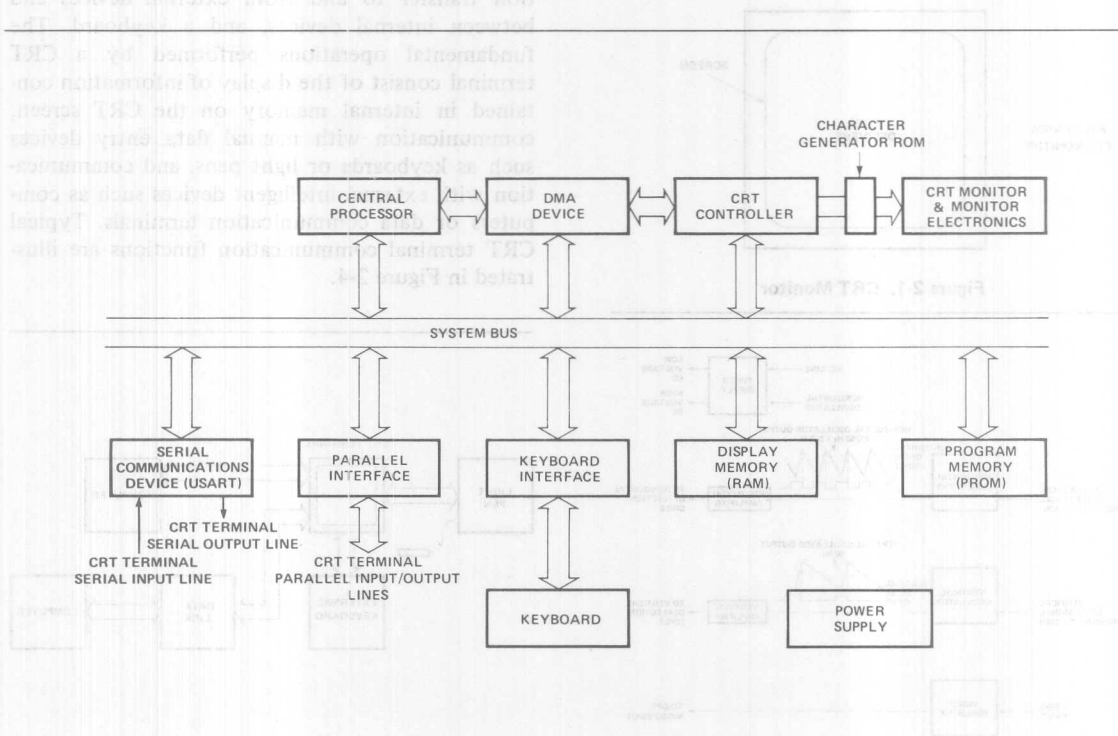


Figure 2-5. CRT Terminal Block Diagram

### 3. COMPONENT DESCRIPTION

#### 3.1 8275

The block diagram and pin configuration for the 8275 Programmable CRT Controller are presented in Figure 3-1. The 8275 provides the following general capabilities:

1. *CRT Display Refreshing* — The 8275, having been programmed to a specific screen format, generates a series of DMA request signals, resulting in the transfer of a row of characters from display memory, via the 8257 DMA Controller, to the 8275's row buffers. The 8275 presents the character codes to an external character generator ROM. The 8275 character code outputs CC0–CC6 are used for this purpose. External dot timing logic is then utilized to transfer the parallel output data from the character generator ROM, serially, to the video input of the CRT. The character rows are displayed on the CRT one line at a time. Line count outputs LC0–LC3 are applied to the character generator ROM to perform the line selection function. The display process is graphically illustrated in Figure 3-2. The entire process is repeated for each display row. At the beginning of the last display row, the 8275 issues an interrupt via the INT output line. The 8275 interrupt output will normally be connected to the interrupt input of the system central processor. The interrupt causes the CPU to execute an interrupt service subroutine. The service subroutine typically re-initializes DMA controller parameters for the next display refresh cycle, polls the system keyboard controller, and/or executes other appropriate functions. A block diagram of a CRT system implemented with the 8275 CRT Controller is provided in Figure 3-3. Proper CRT refreshing requires that certain 8275 parameters be programmed prior to the beginning of display operation. The 8275 has two types of programming registers, the Command Registers (CREG) and the Parameter Registers (PREG). It also has a Status Register (SREG). The Command Registers may only be written to and the Status Registers may only be read. The 8275 expects to receive a command followed by a sequence of from 0 to 4 parameters, depending on the command. The 8275 instruction set consists of 8 commands:

COMMAND	NO. OF PARAMETER BYTES	NOTES
RESET	4	Display format parameters required
START DISPLAY	0	DMA operation parameters included in command
STOP DISPLAY	0	—
READ LIGHT PEN	2	—
LOAD CURSOR	2	Cursor X,Y position parameters required
ENABLE INTERRUPT	0	—
DISABLE INTERRUPT	0	—
PRESET COUNTERS	0	Clears all internal counters

In order to establish the format of the display, the 8275 provides a number of user programmable display format parameters. Display formats having from 1 to 80 characters per row, 1 to 64 rows per screen, and from 1 to 16 horizontal lines per row are available.

In addition to transferring characters from memory to the CRT screen, the 8275 features cursor position control. The cursor position may be programmed, via X and Y cursor position registers, to any character position on the display. The user may select from 4 cursor formats. Blinking or non-blinking underline and reverse video block cursors are available.

2. *CRT Timing* — The 8275 provides two timing outputs, HRTC and VRTC, which are utilized in synchronizing CRT horizontal and vertical oscillators to the 8275 refresh cycle. In addition, whenever HRTC or VRTC are active, a third timing output, VSP (Video Suppress) is true, providing a blanking signal to the dot timing logic. The dot timing logic will normally inhibit the video output to the CRT during the time when video suppress signal is true. An additional timing output, LTEN (Light Enable) is used to provide the ability to force the video output high regardless of the state of VSP. This feature is utilized by

The HGLT (Highlight) output allows an attribute function to increase the CRT beam intensity to a level greater than normal. The fifth timing signal, RVV (Reverse Video) will, when enabled, cause the system video output to be inverted.

### 3. Special Functions —

**VISUAL ATTRIBUTES** — Visual attributes are special codes which, when retrieved from display memory by the 8275, affect the visual characteristics of a character position or field of characters. Two types of visual attributes exist, character attributes and field attributes.

**Character Attribute Codes:** Character attribute codes are codes that can be used to generate graphics symbols without the use of a character generator. This is accomplished by selectively activating the Line Attribute outputs (LA0—LA1), the Video Suppression output (VSP), and the Light Enable output. The dot timing logic uses these signals to generate the proper symbols. Character attributes can be programmed to blink or be highlighted individually. Blinking is accomplished with the Video Suppression output (VSP). Blink frequency is equal to the screen refresh frequency divided by 32. Highlighting is accomplished by activating the Highlight output (HGLT). Character attributes were designed to produce the graphic symbols shown in Figure 3-4.

**Field Attribute Codes:** The field attributes are control codes which affect the visual characteristics for a field of characters, starting at the character following the field attribute code up to, and including, the character which precedes the next field attribute code, or up to the end of the frame.

There are six field attributes:

1. *Blink* — Characters following the code are caused to blink by activating the Video Suppression output (VSP). The blink frequency is equal to the screen refresh frequency divided by 32.
2. *Highlight* — Characters following the

3. *Reverse Video* — Characters following the code are caused to appear in reverse video format by activating the Reverse Video output (RVV).

4. *Underline* — Characters following the code are caused to be underlined by activating the Light Enable output (LTEN).

5. *General Purpose* — There are two additional 8275 outputs which act as general purpose, independently programmable field attributes. These attributes may be used to select colors or perform other desired control functions.

The 8275 can be programmed to provide visible or invisible field attribute characters as shown in Figure 3-5. If the 8275 is programmed in the visible field attribute mode, all field attributes will occupy a position on the screen. They will appear as blanks caused by activation of the Video Suppression output (VSP). The chosen visual attributes are activated after this blanked character. If the 8275 is programmed in the invisible field attribute mode, the 8275 row buffer FIFOs are activated. The FIFOs effectively lengthen the row buffers by 16 characters, making room for up to 16 field attribute characters per display row. The FIFOs are 16 characters by 7 bits in size. When a field attribute is placed in the row buffer during DMA, the buffer input controller recognizes it and places the next character in the proper FIFO. When a field attribute is placed in the buffer output controller during display, it causes the controller to immediately put a character from the FIFO on the Character Code outputs (CC0—6). The chosen attributes are also activated.

**LIGHT PEN DETECTION** — A light pen consists fundamentally of a switch and light sensor. When the light pen is pressed against the CRT screen, the switch enables the light sensor. When the raster sweep coincides with the light sensor position on the display, the light pen output is acti-

and character position coordinates are stored in two 8275 internal registers. These registers can be read on command by the microprocessor.

**SPECIAL CODES** — Four special codes may be used to help reduce memory, software, or DMA overhead. These codes are placed in character positions in display memory.

1. *End of Row Code* —

Activates VSP. VSP remains active until the end of the line is reached. While VSP is active, the screen is blanked.

2. *End of Row-Stop DMA Code* —

Causes the DMA Control Logic to stop DMA for the rest of the row when it is written into the row buffer.

3. *End of Screen Code* —

Activates VSP. VSP remains active until the end of the frame is reached.

4. *End of Screen-Stop DMA Code* —

Causes the DMA Control Logic to stop DMA for the rest of the frame when it is written into the row buffer. It affects the display in the same way as the End of Screen Code.

**PROGRAMMABLE DMA BURST CONTROL** —

The 8275 can be programmed to request single byte DMA transfers or DMA burst transfers of 2, 4, or 8 characters per burst. The interval between bursts is also programmable. This allows the user to tailor his DMA overhead to fit his system needs.

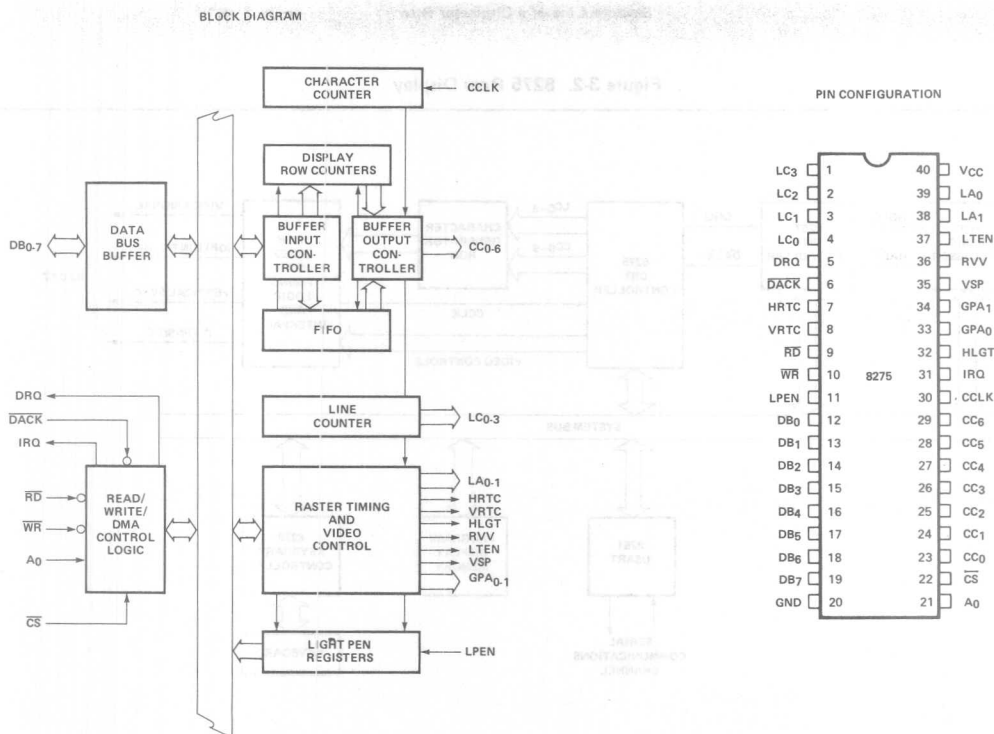


Figure 3-1. 8275 Block Diagram/Pin Configuration

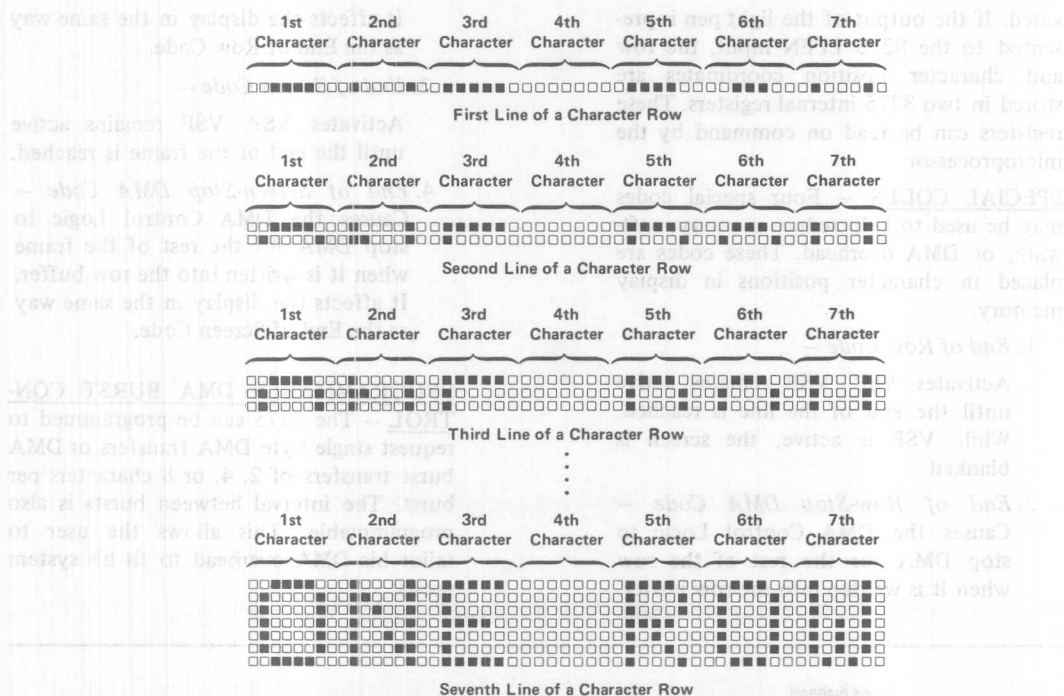


Figure 3-2. 8275 Row Display

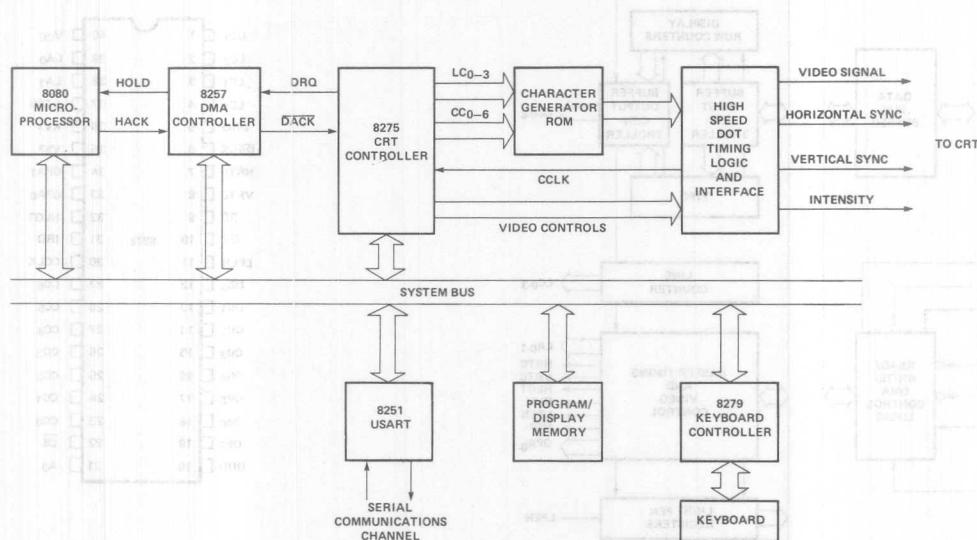


Figure 3-3. CRT System Block Diagram

Character attributes were designed to produce the following graphics:

CHARACTER ATTRIBUTE CODE "CCCC"	OUTPUTS				SYMBOL	DESCRIPTION
	LA <sub>1</sub>	LA <sub>0</sub>	VSP	LTEN		
0000	Above Underline	0	0	1	0	Top Left Corner
	Underline	1	0	0	0	
	Below Underline	0	1	0	0	
0001	Above Underline	0	0	1	0	Top Right Corner
	Underline	1	1	0	0	
	Below Underline	0	1	0	0	
0010	Above Underline	0	1	0	0	Bottom Left Corner
	Underline	1	0	0	0	
	Below Underline	0	0	1	0	
0011	Above Underline	0	1	0	0	Bottom Right Corner
	Underline	1	1	0	0	
	Below Underline	0	0	1	0	
0100	Above Underline	0	0	1	0	Top Intersect
	Underline	0	0	0	1	
	Below Underline	0	1	0	0	
0101	Above Underline	0	1	0	0	Right Intersect
	Underline	1	1	0	0	
	Below Underline	0	1	0	0	
0110	Above Underline	0	1	0	0	Left Intersect
	Underline	1	0	0	0	
	Below Underline	0	1	0	0	
0111	Above Underline	0	1	0	0	Bottom Intersect
	Underline	0	0	0	1	
	Below Underline	0	0	1	0	
1000	Above Underline	0	0	1	0	Horizontal Line
	Underline	0	0	0	1	
	Below Underline	0	0	1	0	
1001	Above Underline	0	1	0	0	Vertical Line
	Underline	0	1	0	0	
	Below Underline	0	1	0	0	
1010	Above Underline	0	1	0	0	Crossed Lines
	Underline	0	0	0	1	
	Below Underline	0	1	0	0	
1011	Above Underline	0	0	0	0	Not Recommended *
	Underline	0	0	0	0	
	Below Underline	0	0	0	0	
1100	Above Underline	0	0	1	0	Special Codes
	Underline	0	0	1	0	
	Below Underline	0	0	1	0	
1101	Above Underline					Illegal
	Underline		Undefined			
	Below Underline					
1110	Above Underline					Illegal
	Underline		Undefined			
	Below Underline					
1111	Above Underline					Illegal
	Underline		Undefined			
	Below Underline					

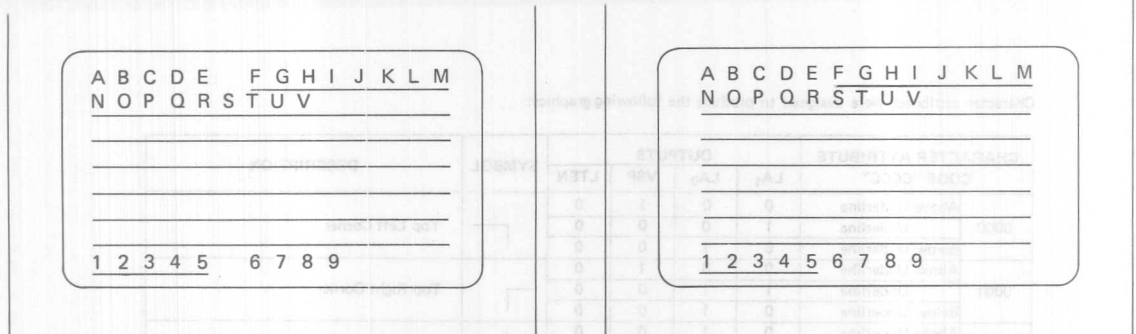
\*Character Attribute Code 1011 is not recommended for normal operation. Since none of the attribute outputs are active, the character generator will not be disabled, and an indeterminate character will be generated.

Character Attribute Codes 1101, 1110, and 1111 are illegal.

Blinking is active when B = 1.

Highlight is active when H = 1.

Figure 3-4. Character Attributes



EXAMPLE OF THE VISIBLE FIELD ATTRIBUTE MODE  
(UNDERLINE ATTRIBUTE)

EXAMPLE OF THE INVISIBLE FIELD ATTRIBUTE MODE  
(UNDERLINE ATTRIBUTE)

Figure 3-5. Field Attribute Examples

### 3.2 8279

The 8279 Programmable Keyboard/Display Interface block diagram and pin configuration are shown in Figure 3-6. The 8279 will be utilized in the CRT design example for performing keyboard scanning, key debounce, and data bus interface functions. Only features associated with these

functions will be described in this section. The reader is referred to the 8279 data sheet for information on display control, sensor matrix mode operation, and strobed input mode operation. A detailed description of the 8279 keyboard scanning, debounce, and data bus interface functions follows.

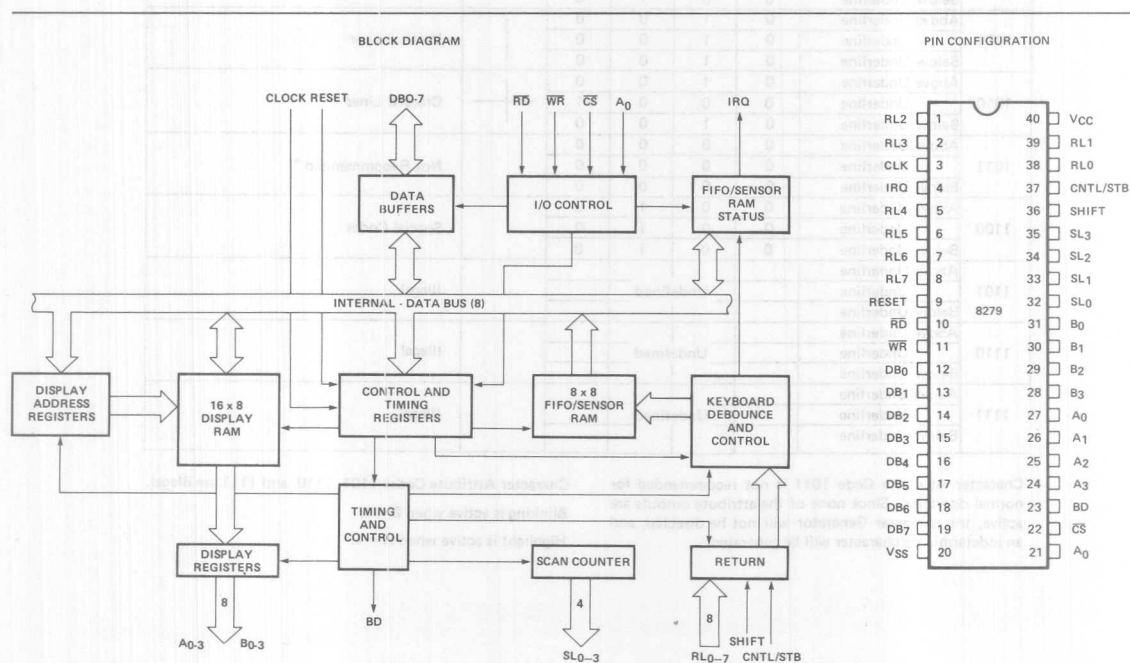


Figure 3-6. 8279 Pin Configuration and Block Diagram

The primary functions of the 8279 in the CRT system application include scanning the 64 key keyboard, determining if a key has been depressed, and, when polled by the system processor, transmitting the address of the key in the keyboard matrix to the master processor. Alternately, the interrupt line from the 8279 may be used to inform the CPU of a key depression. A block diagram of the 8279 interface, as implemented in the CRT system design example, is provided in Figure 3-7. The keyboard controller initiates the keyboard scanning process by transmitting keyboard scan line selection information over output lines SL<sub>0</sub>–SL<sub>2</sub>. The data may be encoded or decoded depending on the mode programmed. Assuming encoded mode is selected, the SL<sub>0</sub>–SL<sub>2</sub> lines are connected to the input of a 3-line to 8-line decoder as shown in Figure 3-7. The decoder outputs are connected to the keyboard row inputs. Only one decoder output will be enabled for a given set of input conditions. The keyboard column outputs are connected to the 8279 return line inputs RL<sub>0</sub>–RL<sub>7</sub>. The eight return lines are buffered and latched by the 8279. These lines are scanned by the internal logic of the 8279, looking for a key depression in the selected row. If the debounce circuit detects a key depression, it waits approximately 10 ms to determine if the key remains down. If it does, the address of the key in the matrix plus the status of the shift and control lines are transferred to the 8279 FIFO. The FIFO data format is shown in Figure 3-8. The FIFO will hold up to eight data bytes; that is, up to eight key depressions may occur prior to a CPU initiated read operation. The number of characters entered into the FIFO is indicated by the character count contained within the FIFO status word. When a key depression is detected, the 8279 interrupt line goes high, and the FIFO status is modified to reflect the number of characters contained in the FIFO. The CPU may determine the occurrence of a key depression in one of two ways: The 8279 interrupt line may be connected to the interrupt input line of the CPU, forcing the CPU to call an interrupt service routine which reads the FIFO character. An alternate approach requires the CPU to periodically poll the 8279, reading the FIFO status word. If the FIFO character count is non-zero, indicating that at least one character is present in the FIFO, the CPU then reads the FIFO contents. This approach will be utilized in the CRT design example. A read operation places the contents of the FIFO on the system data bus and decrements the FIFO character

count, contained within the FIFO status word, by one.

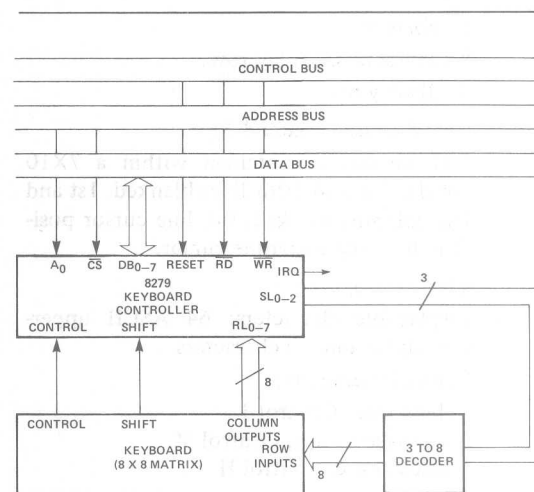


Figure 3-7. 8279 Interface

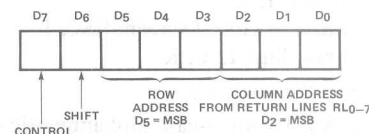


Figure 3-8. FIFO Data Byte Format

## 4. CRT SYSTEM DESIGN EXAMPLE

### 4.1 SCOPE OF THE PROJECT

A fully operational, microcomputer-based CRT terminal was designed and constructed utilizing the 8275 CRT Controller and 8279 Keyboard Controller as the basic system elements. The terminal incorporates the majority of the functions found in existing dedicated computer terminals. An Intel® 8080A microprocessor was utilized as the CPU in the design. The recently announced Intel® 8085 microprocessor constitutes an ideal processor for future CRT terminal designs. LSI devices were utilized in the design whenever possible in order to minimize component count.

## 4.2 SYSTEM SPECIFICATIONS

The specifications for the CRT terminal design are as follows:

### Display Format

- 80 characters/display row
- 25 display rows

### Character Format (Figure 4-1)

- 5X7 character contained within a 7X10 matrix, 1st and 10th lines blanked, 1st and 7th columns blanked, 9th line cursor position, blinking underline cursor.

### Characters Recognized

- Displayable characters: 64 ASCII upper-case alphanumeric characters
- Control characters:
  - Line feed, Control J
  - Carriage return, Control M
  - Back space, Control H
- Escape Sequences:
  - Cursor up, ESC, A
  - Cursor down, ESC, B
  - Cursor right, ESC, C
  - Cursor left, ESC, D
  - Clear screen, ESC, E
  - Home, ESC, H
  - Erase to end of screen, ESC, J
  - Erase line, ESC, K

### Characters Transmitted

- 64 ASCII upper-case alphanumeric characters
- ASCII Control Character set
- ASCII Escape Sequence set

### Program Memory

- 2K bytes, 2716 EPROM

### Display/Buffer/Stack Memory

- 2K bytes, 2114 static RAM

### Data Rate

- 4800 BAUD maximum using 8080A

### CRT Monitor

- Ball Bros TV-12, 12 MHz B.W.

### Keyboard

- Microswitch hall effect keyboard, open collector outputs

### Scrolling Capability

- Scroll up feature implemented with 8257 DMA Controller

### Screen Refresh Rate

- 60 Hz

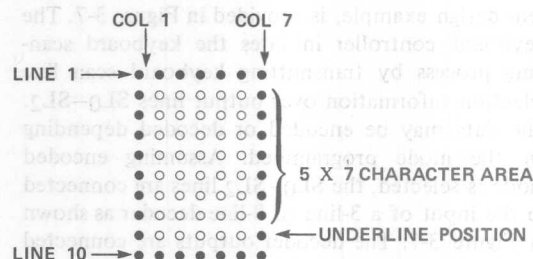


Figure 4-1. Character Format

## 4.3 SYSTEM HARDWARE DESIGN

### 4.3.1 General Considerations

A block diagram of the CRT terminal is presented in Figure 4-2. The diagram includes only essential system features. A detailed schematic of the CRT terminal is contained in the appendix. The terminal was constructed using an Intel® SDK-80 microcomputer kit and an Intel® SBC 905 prototyping board. The standard 8080 bus structure incorporated in the SDK-80 kit allowed the CRT terminal to be implemented with minimum buffering.

In the ensuing discussion of CRT terminal operation, it will be assumed that the terminal normally communicates with a remote device, such as an Intel® MDS microcomputer development system. Communication will take place in the full duplex mode. The CRT terminal, upon transmitting a character to the remote device, will remain idle until a character is received from the external device. Transmission of a character to the remote device is initiated by depressing a key on the keyboard. Character transmission to the CRT terminal from the remote device is assumed to be asynchronous with respect to terminal operation.



rate necessary for CRT refreshing. Display characters are then transferred from the 8275 row buffers to the character code outputs CC0–CC5. The character code outputs are applied to the character generator address lines A3–A8 (Figure 4-3). Line count outputs LC0–LC2 from the 8275 are applied to character generator address lines A0–A2. It should be noted that the 8275 displays character rows one line at a time. The line count outputs are utilized to determine which line of the character selected by A3–A8 will be displayed. Following the transfer of the first line to the dot timing logic, the line count is incremented and the second line of the character row is selected. The process continues until the last line of the row under consideration is transferred to the dot timing logic.

The dot timing logic latches the 6-bit character code and 3-bit line count from the 8275 on positive transitions of the character clock and transfers this information to the character generator ROM. In systems requiring a greater number of lines/character, the fourth line count output would also be used. The 7-bit ROM output corresponds to the 7 dots which make up a line segment for a particular character. The ROM output is loaded into a parallel input-serial output shift register. The shift register is clocked at the dot clock rate (11.34 MHz) continuously. The shift register output constitutes the video input to the CRT. The character code outputs select the character to be displayed at a given character position in the display row. The character set consists of  $2^6=64$  ASCII upper case alphanumeric characters.

The row by row transfer of character data from display memory to the 8275 continues until the beginning of the last display row. At this time the 8275 issues an interrupt to the CPU. The CPU polls both the 8275 and 8251. Having determined that the interrupt originated with the 8275, the CPU calls the 8275 interrupt subroutine. The 8275 interrupt subroutine re-initializes the 8257 DMA Controller starting address and terminal count parameters and polls the 8279 Keyboard Controller to determine if a key depression has occurred. If a key has been depressed, the CPU reads the key position data from the 8279, performs a table lookup, and transmits the appropriate ASCII character to the CRT data output via the 8251 USART. It should be noted that interrupts are generated by the 8275 every 16.67 ms for a 60 Hz screen refresh rate.

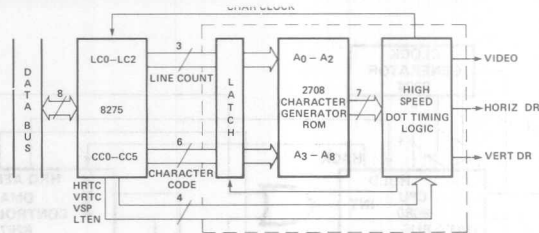


Figure 4-3. Character Generator/Dot Timing Logic Block Diagram

### 4.3.3 System Timing

The CRT terminal display raster is shown in Figure 4-4. It can be seen from the figure that a display row is composed of 10 lines. The Total Line Time consists of the display portion of the line plus the Horizontal Blanking Time. Row Time is equal to the number of lines per row multiplied by the Total Line Time. The Total Screen Time (1/Refresh Rate) is equal to the Row Time multiplied by the number of display rows plus the Row Time intervals associated with vertical blanking. Specifications for the BALL BROS. monitor show that there are constraints on the Vertical Blanking Time, Horizontal Blanking Time, and Horizontal Oscillator Repetition Rate. These constraints are summarized in Table 4-1.

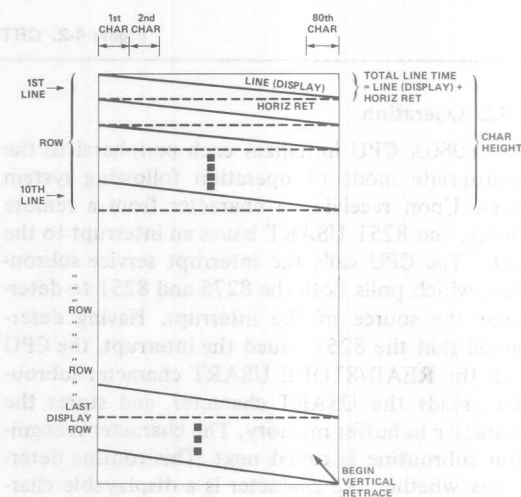


Figure 4-4. CRT Display Raster

Table 4-1

PARAMETER	RANGE
Vertical Blanking Time (VRTC)	900 $\mu$ sec nominal
Vertical Drive Pulsewidth	300 $\mu$ sec $\leq$ PW $\leq$ 1.4 ms
Horizontal Blanking Time (HRTC)	11 $\mu$ sec nominal
Horizontal Drive Pulsewidth	25 $\mu$ sec $\leq$ PW $\leq$ 30 $\mu$ sec
Horizontal Repetition Rate	15,750 $\pm$ 500 pps

Given the constraints in Table 4-1 and the Refresh Rate specification of 60 Hz, the Vertical Retrace Row Count and Horizontal Retrace Character Count parameters required by the 8275 CRT Controller may be calculated:

$$\begin{aligned}\text{Total Screen Time} &= \frac{1}{\text{Refresh rate}} = \frac{1}{60 \text{ Hz}} \\ &= 0.01667 \text{ sec}\end{aligned}$$

Also,

$$\begin{aligned}\text{Total Screen Time} &= (\text{Row Time}) (\# \text{ of Display Rows}) \\ &+ \text{Vertical Blanking Time (VRTC)}\end{aligned}$$

Vertical Blanking Time (VRTC) must be an integral number of Row Times (between 1 and 4).

Therefore,

$$\begin{aligned}0.016667 \text{ sec} &= (\text{Row Time}) (25) + \text{VRTC} \\ &= (\text{Row Time}) (25) + N (\text{Row Time})\end{aligned}$$

If N is selected to be 2, the following result is obtained:

$$\text{Row Time} = 6.17284 \times 10^{-4} \text{ sec}$$

Therefore,

$$\begin{aligned}\text{VRTC} &= (2)(\text{Row Time}) = 12.3457 \times 10^{-4} \text{ sec} \\ &= 1.23457 \text{ ms}\end{aligned}$$

Since the Vertical Blanking Time, nominally 900  $\mu$ sec, falls within the constraints for the Vertical Drive Pulsewidth, the VRTC output from the 8275 may be used directly for the Vertical Drive Pulse. The 8275 will be programmed for a Vertical Retrace Row Count of 2.

In order to calculate the Horizontal Retrace Character Count, it is necessary to consider the row format as defined in the specifications. Figure 4-5

shows three adjacent characters in a row. The row, as shown, is composed of 10 Lines/Row and 7 Dots/Line/Character. Given that the Row Time is 617.284  $\mu$ sec, the Total Line Time may be calculated as follows:

$$\begin{aligned}\text{Total Line Time} &= \frac{\text{Row Time}}{\# \text{ Lines/Row}} \\ &= \frac{617.284 \times 10^{-6} \text{ sec}}{10} \\ &= 61.7284 \times 10^{-6} \text{ sec} \\ &= 61.7284 \mu\text{sec}\end{aligned}$$

The Total Line Time is composed of the display portion of the line plus the Horizontal Blanking Time (HRTC).

$$\begin{aligned}\text{Total Line Time} &= 61.7284 \times 10^{-6} \text{ sec} \\ &= 80 \left( \frac{\text{Character Time}}{\text{line}} \right) + \text{HRTC}\end{aligned}$$

Horizontal Blanking Time (HRTC) must be an integral number of Character Times/Line.

Then

$$\begin{aligned}61.7284 \times 10^{-6} \text{ sec} &= 80 \left( \frac{\text{Character Time}}{\text{line}} \right) \\ &+ M \left( \frac{\text{Character Time}}{\text{line}} \right)\end{aligned}$$

If M is selected to be 20, the following result is obtained:

$$\begin{aligned}\left( \frac{\text{Character Time}}{\text{line}} \right) &= \frac{61.7284 \times 10^{-6}}{80 + 20} \\ &= 6.1728 \times 10^{-7} \text{ sec} \\ &= 617.284 \text{ ns}\end{aligned}$$

This value defines the period of the 8275 character clock.

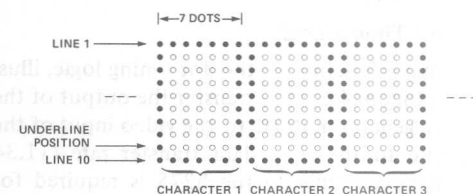


Figure 4-5. Row Format

The Horizontal Blanking Time (HRTC) is calculated as follows:

$$\begin{aligned} \text{HRTC} &= 20 (617,284 \text{ ns}) \\ &= 12.3456 \mu\text{sec} \text{ (nominal value } 11 \mu\text{sec)} \end{aligned}$$

The 8275 will be programmed for a Horizontal Retrace Character Count of 20. Since the specifications call for a Horizontal Drive Pulsewidth of 25–30  $\mu\text{sec}$ , an external oneshot is required. The oneshot is triggered by the leading edge of HRTC.

Using the value for the Character Time/Line, the Dot Clock Rate may be established. It should be noted that the clock is used to shift data from the parallel in-serial out shift register (contained in the dot timing logic) to the CRT video input. The system character clock is also derived from the Dot Clock.

The dot clock is calculated as follows:

$$\begin{aligned} \left( \frac{\text{Dot Time}}{\text{line}} \right) &= \left( \frac{\text{Character Time}}{\text{line}} \right) \frac{\# \text{ dots/character}}{\text{line}} \\ &= \frac{6.17284 \times 10^{-7}}{7} \text{ sec} \\ &= 8.8183 \times 10^{-8} \text{ sec} \\ &= 88.183 \text{ ns} \end{aligned}$$

$$\text{Dot Clock Frequency} = \frac{1}{\frac{\text{Dot Time}}{\text{Line}}} = 11.34 \text{ MHz}$$

The Horizontal Oscillator Repetition Rate may be calculated as follows:

$$\begin{aligned} f_{\text{Horiz}} &= \frac{1}{\text{Total Line Time}} = \frac{1}{61.7284 \times 10^{-6} \text{ sec}} \\ &= 16,200 \text{ Hz} \end{aligned}$$

This value falls within the system specification of  $15,750 \pm 500 \text{ pps}$ .

#### 4.3.4 Dot Timing Logic

The primary function of the dot timing logic, illustrated in Figure 4-6, is to transfer the output of the character generator ROM to the video input of the CRT. Due to the high data transfer rate (11.34 MHz), logic external to the 8275 is required for this function. The data transfer operation is accomplished as follows: The character generator

ROM output is applied to the parallel input lines of the 74166 shift register, the shift register is loaded synchronously with respect to the positive-going edge of the character clock, and data is clocked out of the 74166 serial input at the dot clock frequency. The 74166 output is applied, through appropriate gating logic, to the CRT video input. In addition to the previously described functions, the dot timing logic provides the timing signals required for transferring characters from the 8275 character code and line count outputs to the character generator ROM, implements the video suppress and light enable gating functions, and generates the system dot and character clocks.

In order to understand the dot timing logic design process, it is necessary to refer to Figure 4-6 and Figure 4-7.

It can be seen from the timing waveforms of Figure 4-7 that the character code output from the 8275 will be valid 150 ns (worst case) after the negative-going edge of the character clock. The character generator ROM output will be valid, assuming a direct connection between the 8275 and the ROM, 450 ns (worst case) after the character code appears at the address inputs. Total delay from the negative-going edge of the character clock until ROM output data becomes available is then 600 ns. Given the character clock width of 617 ns and external logic propagation delays and setup times, it becomes difficult to latch the ROM output for the first display character during the first character clock period. In order to alleviate this situation, a data pipelining technique is utilized. The timing for this technique is shown in Figure 4-7. A latch, introduced between the 8275 and the character generator ROM as shown in Figure 4-6, samples character code and line count data from the 8275 1/2 dot clock (45 ns) after the positive-going edge of the character clock. Data from the latch is applied to the character generator ROM address lines yielding, after a 450 ns delay (worst case), the appropriate 7-bit code at the ROM output. ROM data is loaded into the 74166 shift register on the next positive-going edge of the character clock. This technique effectively delays the video output from the shift register by 1½ character clocks, but eliminates the difficulties in sampling the ROM data within the first character clock period. Due to the video delay associated with this technique, it is also necessary to delay all signals affecting the video output and CRT timing. These signals include HRTC, VRTC, VSP, and

LTEN. The delay is accomplished using a two-stage shift register constructed with edge triggered D flip-flops (74175). The system dot clock (11.34 MHz) is obtained by dividing the 22.68 MHz output from the 8224 clock generator by two. The dot clock is utilized to clock the 74166 output shift register

and is divided by 7, using a 74S163 counter, to produce the system character clock. It should be noted that the use of a bipolar character generator PROM such as the Intel® 3604 or 3608 will reduce the external dot timing logic package count due to the reduced access time.

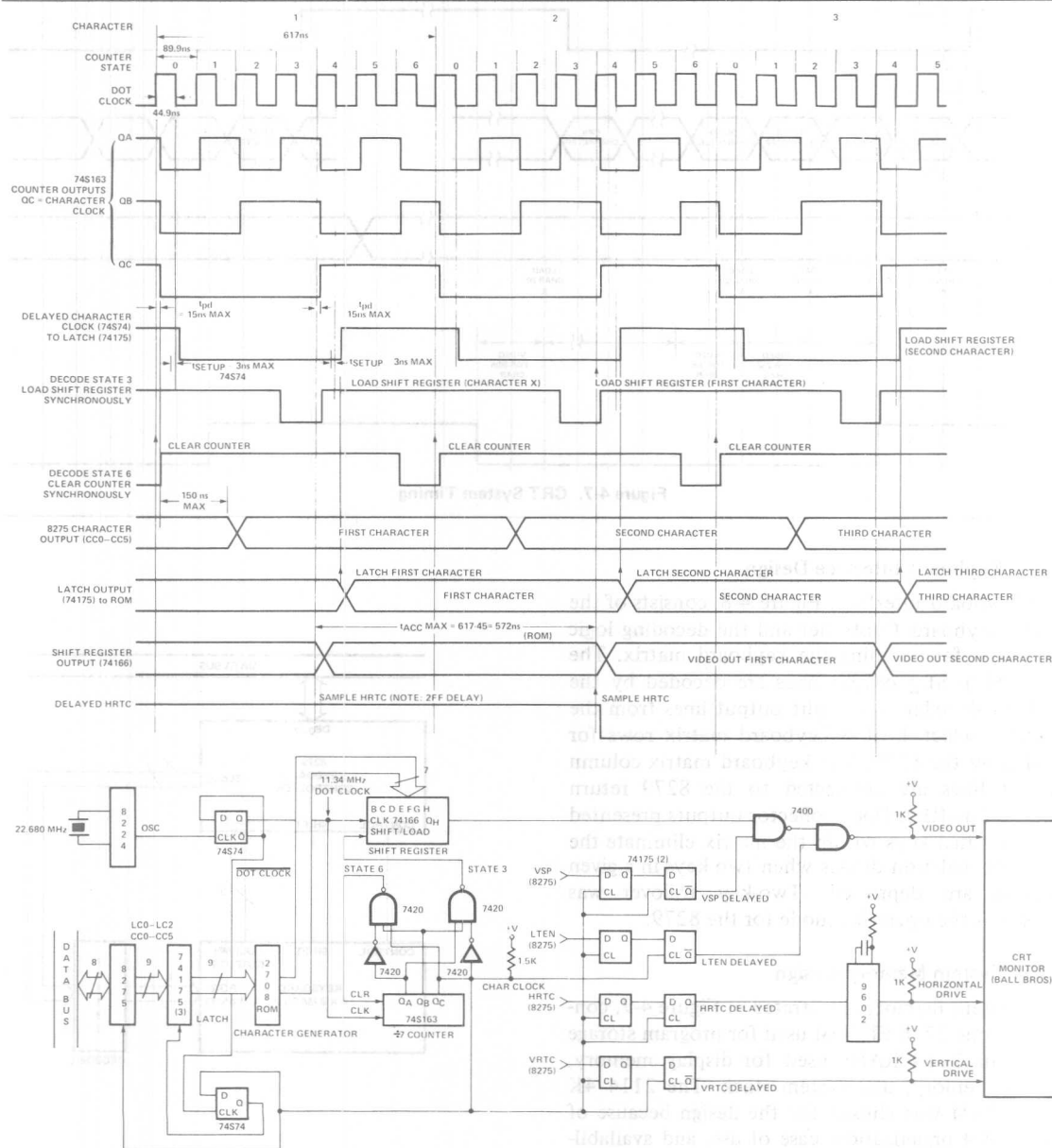


Figure 4-6. Dot Timing Logic

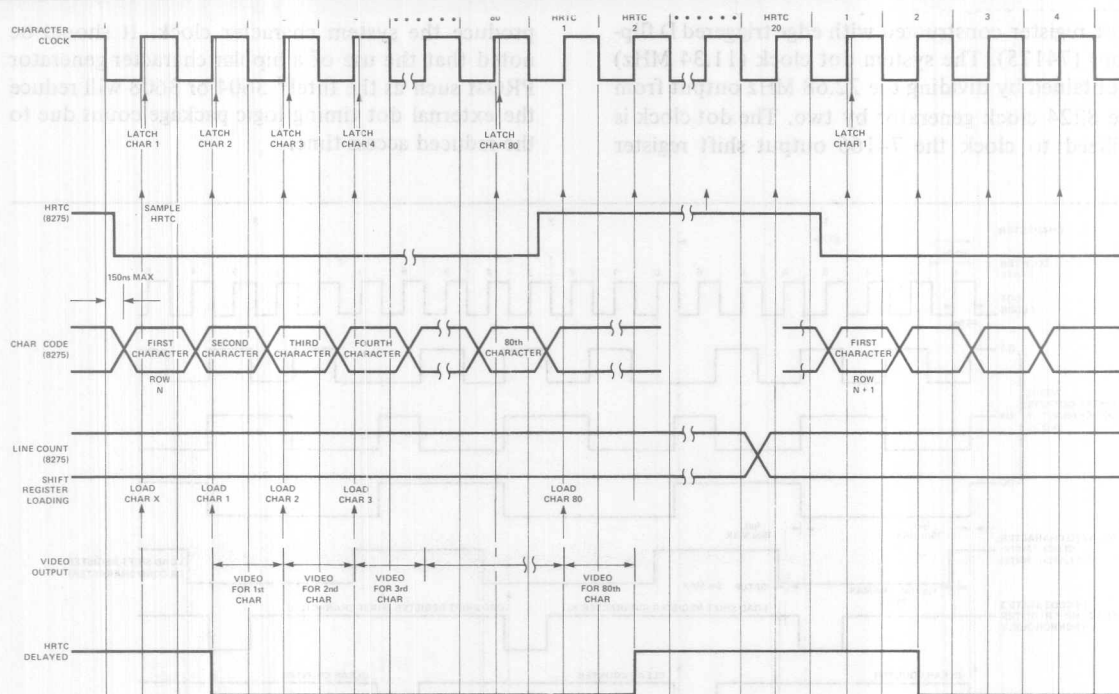


Figure 4-7. CRT System Timing

### 4.3.5 Keyboard Interface Design

The keyboard interface, Figure 4-8, consists of the 8279 Keyboard Controller and the decoding logic necessary for scanning the keyboard matrix. The 8279  $SL_0$ – $SL_2$  output lines are decoded by the 74S138 decoder. The eight output lines from the decoder select 1 of 8 keyboard matrix rows for testing by the 8279. The keyboard matrix column output lines are connected to the 8279 return lines,  $RL_0$ – $RL_7$ . Open collector outputs presented by individual keys within the matrix eliminate the need for isolation diodes when two keys in a given column are depressed. Two-key rollover was chosen as the operating mode for the 8279.

### 4.3.6 System Memory Design

The system memory, illustrated in Figure 4-9, consists of one 2716 EPROM used for program storage and four 2114 RAMs used for display memory, buffer memory, and system stack. The 2114 4K static RAM was chosen for the design because of its  $1K \times 4$  organization, ease of use, and availability. Buffering between RAM memory and the system data bus was used to minimize bus loading.

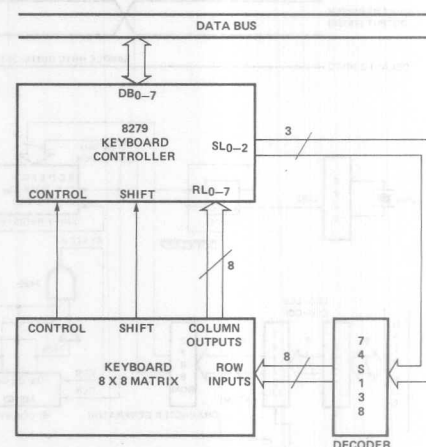


Figure 4-8. Keyboard Interface



system reset, the central processor interrupt system is disabled, the program counter is set to zero, and peripheral reset functions are carried out. Following reset, the system software initializes all peripherals, clears buffer memory, initializes special buffer locations, fills display memory with space codes, and enables interrupts. The processor then loops until an interrupt arrives from the 8275 or 8251. When the processor detects the occurrence of an interrupt, the instruction being executed is completed, an RST 7 vector is placed on the system data bus, and the RST 7 call instruction is executed, forcing a jump to the starting address of the 8275/8251 interrupt polling routine. Once the polling routine establishes the source of the interrupt, program flow continues along one of the two possible paths shown in Figure 4-10. An 8275 interrupt causes the 8257 DMA Controller to be re-initialized, the 8279 Keyboard Controller to be serviced, and, if a key depression has occurred, a character to be transmitted to the terminal output. An interrupt from the 8251 will first cause the USART character to be read and stored in mem-

ory. The system software then examines the character to determine whether it is a displayable character, a control code, or the first or second character in an escape sequence. After determining the nature of the character, an appropriate subroutine is called. Following the completion of the routines associated with an 8275/8251 interrupt, interrupts are re-enabled and a return instruction executed. The CPU then loops until the receipt of an interrupt. In order to appreciate the operation of the system software in detail, it is necessary to consider the following items:

1. System memory organization.
2. The relationship between character position on the screen and screen pointers Row Count, Column Count, and memory pointer Top.
3. The relationship between memory pointers Row Count, Column Count and the 8275 cursor X and Y position registers.
4. Scrolling concepts, including the relation between scrolling, display memory, and the memory pointer Top.

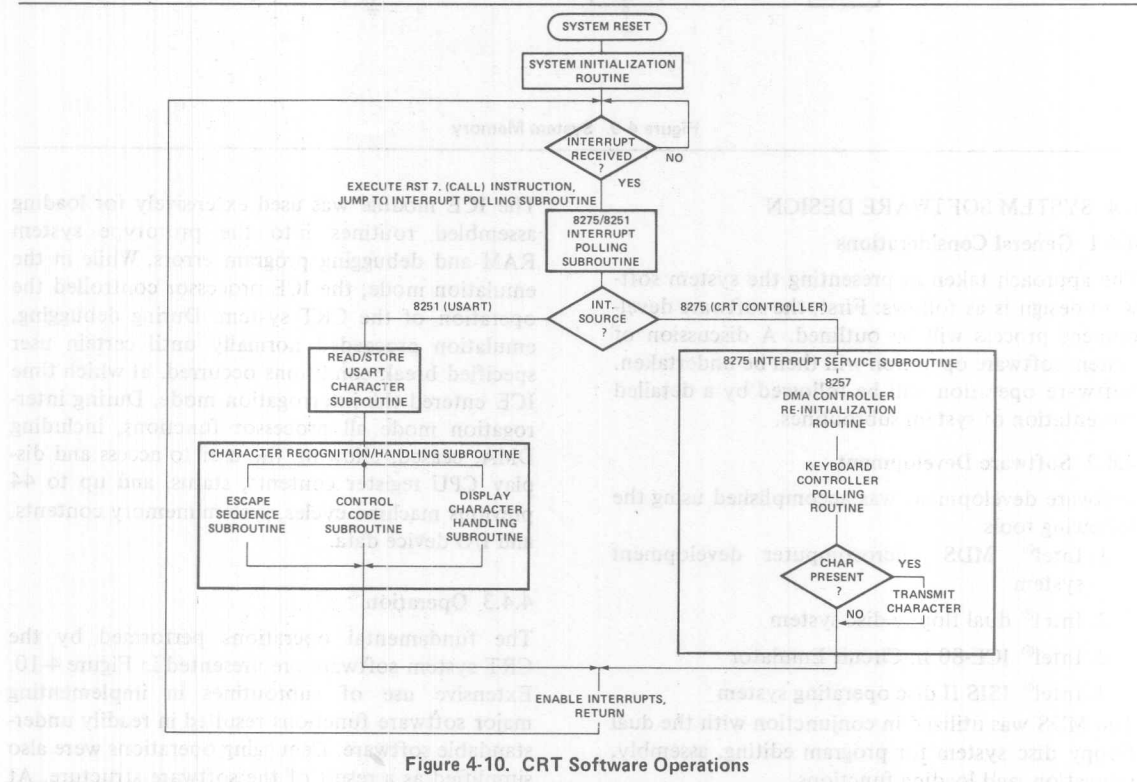


Figure 4-10. CRT Software Operations

### System Memory Organization

System memory organization is shown in Figure 4-11. It should be noted that an additional 2K block of RAM was utilized for program memory (rather than PROM) during the software development/debug phase of system design.

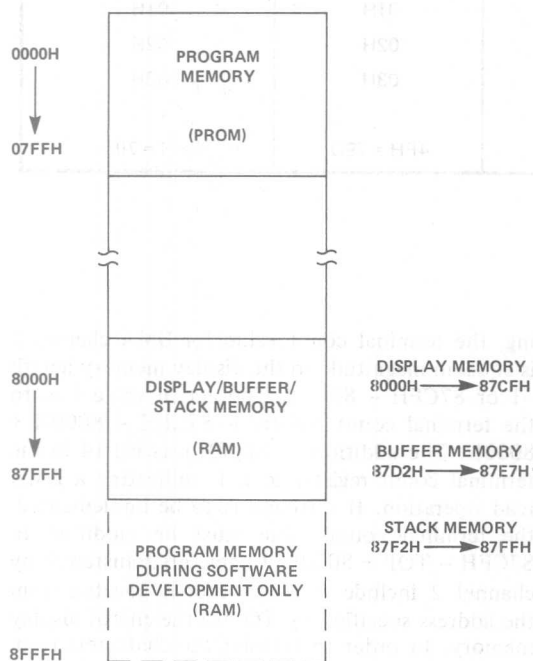


Figure 4-11. System Memory Organization

### Character Position/Screen Pointer Relationships

To define the location of a character on the screen, two pointers, Row Count and Column Count, were created in memory. The relationship between character location on the screen and the two pointers is illustrated in Figure 4-12. Row Count and Column Count are stored in memory locations RCTAD and CCTAD, respectively. Row Count represents the position of the first character in a given row. For the first row, Row Count = 0000H. For the second row, Row Count = 0050H. Column Count represents the specific column in which the character is located. Character position on the screen may be calculated by adding the Row Count to the Column Count; e.g., the highlighted character in Figure 4-12 is located at  $AOH + 03H = A3H$ .

		CRT DISPLAY											
COLUMN		1	2	3	4								80
COLUMN COUNT		00H	01H	02H	03H	•	•	•	•	•	•	•	4FH
		=00D	=01D	=02D	=03D								=79D
ROW	ROW COUNT												
1	0000H = 0000D	0	1	2	3								4F
2	0050H = 0080D	50	51	52	53								9F
3	00A0H = 0160D	A0	A1	A2	A3								EF
4	00F0H = 0240D	F0	F1	F2	F3								13F
	•												
	•												
	•												
	•												
	•												
25	0780H = 1920D	780	781	782	783								7C

**Figure 4-12. Character Location/Pointer Relationship**

### Memory Pointer/8275 Cursor Position Register Relationship

It was necessary to establish a relationship between Row Count and Column Count pointers and the 8275 Cursor X and Y Position registers for the cursor generated by the 8275 to be loaded at the appropriate position on the screen. This relationship is summarized in Table 4-2.

The value transferred to the 8275 for the Cursor X Position is identical to the Column Count. A new parameter, Cursor Y Position, stored at memory location CURSY, was also established. For a given Row Count value, a value for Cursor Y Position is defined. This value is transferred to the 8275 Cursor Y Position register.

It is necessary to introduce an additional parameter, **Top**, which will be used in conjunction with **Row Count** and **Column Count** to determine the location in display memory at which an incoming display character will be stored. The location at which a given character will be stored (assuming no more than 2000 characters have been entered since initialization) is calculated by adding **TOP + Row Count + Column Count**, where **TOP** is assumed to be 8000H, the starting location of display memory shown in Figure 4-11. Following system initialization, characters will be entered in display memory starting at memory location 8000H. The 2000th character will be entered at location 87CFH. Upon entering the 2001st character, a scrolling condition exists and **TOP** will be modified to point to memory address 8050H. An in-depth discussion of scrolling is presented in the next section.

Table 4-2

## SCREEN POINTER/8275 CURSOR X,Y POSITION REGISTER RELATIONSHIP

ROW	ROW COUNT VALUE	CURSOR Y POSITION REGISTER VALUE	COLUMN	COLUMN COUNT VALUE	CURSOR X POSITION REGISTER VALUE
1	0000H	00H	1	00H	00H
2	0050H	01H	2	01H	01H
3	00A0H	02H	3	02H	02H
4	00F0H	03H	4	03H	03H
25	0780H = 1920D	18H = 24D	80	4FH = 79D	4FH = 79D

### Scrolling

Scrolling is implemented in the CRT system design by shifting the entire display up by 1 row when a scrolling condition occurs. Scrolling will occur when certain cursor manipulation functions are exercised or when a character is entered in the last CRT display position, indicating a full memory page condition exists. Character entry will be used as the vehicle for explaining scrolling in the following discussion.

Characters are normally entered sequentially in display memory. When the 2000th character has been entered, display memory capacity has been attained; i.e., a full page condition exists. At this point, scrolling will take place. For scrolling to take place, DMA channel 2, the channel used to extract characters from display memory, must be re-initialized to the appropriate starting address and terminal count values. The memory pointer TOP will be used to establish the starting address for channel 2. Prior to scrolling, TOP = 8000H, the starting address of display memory. Each scrolling operation causes 80D (50H) to be added to TOP, moving the pointer, as shown in Figure 4-13b, to the beginning of the following row in display memory. It should be recalled that TOP, in conjunction with Row Count and Column Count determines the insertion address for incoming display characters. The net effect of modifying TOP is to shift the information being displayed on the CRT up by 1 row; i.e., scrolling is accomplished. Prior to scroll-

ing, the terminal count value for DMA channel 2 is equal in magnitude to the display memory length -1 or 87CFH - 8000H. The actual value sent to the terminal count register is 87CFH - 8000H + 8000H. The addition of 8000H sets bit 14 in the terminal count register to a 1, indicating a DMA read operation. If scrolling is to be implemented, the terminal count value must be modified to 87CFH - TOP + 8000H. Characters transferred by channel 2 include those characters located from the address specified by TOP to the end of display memory. In order to transfer the characters from the beginning of display memory through the address immediately prior to TOP, the autoloading feature of the 8257 DMA controller is utilized. When DMA channel 2 reaches terminal count, following the transfer of characters from TOP to the end of display memory, the starting address and terminal count parameters stored in the DMA channel 3 registers are loaded into channel 2. DMA operations resume in channel 2 using the channel 3 parameters. To accomplish the desired channel 3 operations, it is only necessary to re-initialize the channel 3 starting address to the beginning address of display memory, and the terminal count value to 87CFH, the maximum terminal count for a 2000-byte display memory space. These processes are performed during DMA re-initialization following an 8275 interrupt. New text entry following scrolling is illustrated in Figure 4-13. BOTTOM, a parameter corresponding to the address of the first character in the last row to be displayed, is utilized during clear to end of screen operations.

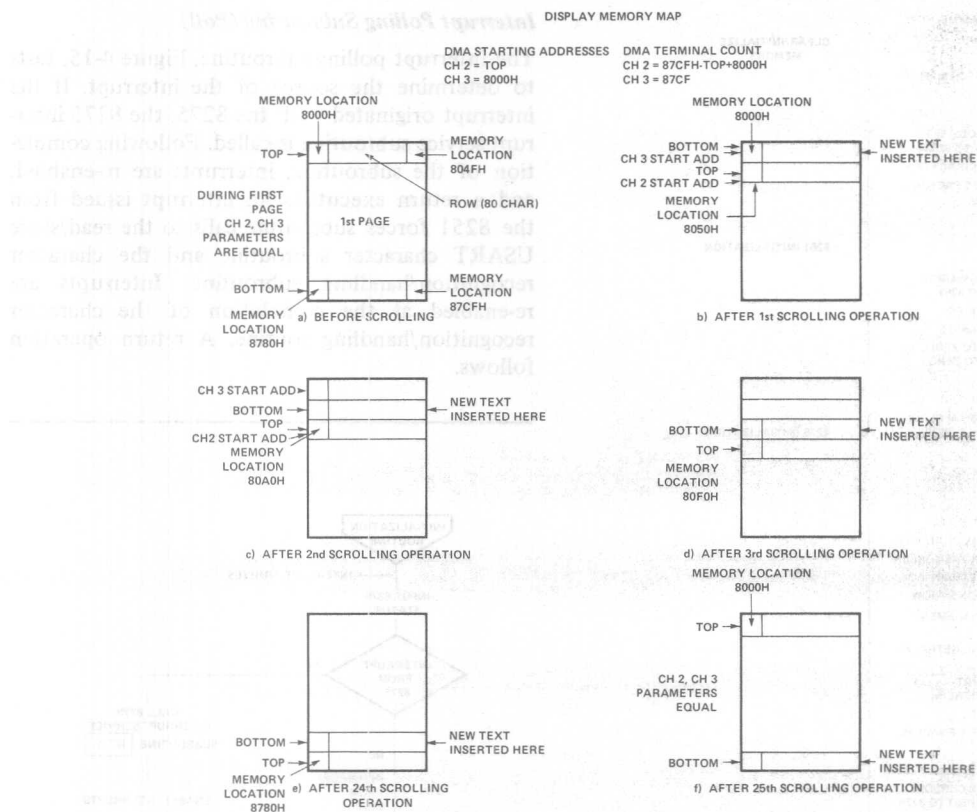


Figure 4-13. Pointer Manipulation During Scrolling

#### 4.4.4 System Subroutines

##### System Initialization Routine (CRTGO)

The system initialization routine, Figure 4-14, establishes a starting point for system operation. The 8251 USART is initialized to transmit to and receive characters from an external device. The 8279 Keyboard Controller, at system reset, comes up in the two-key rollover mode. It is therefore only necessary to set up the Keyboard Controller internal operating frequency during initialization. Assuming a desired internal operating frequency of approximately 100 kHz and a 2.048 MHz system clock, the frequency divider chain is programmed to divide by 21. The 8275 initialization parameters are determined from the original CRT system specifications and vertical retrace Row Count/Horizontal Retrace/Character Count calculations previously performed. The delayed line number feature allows the use of only 3 line count outputs

to determine which of 10 possible lines in a character row will be displayed. Given that the underline placement position is set to the ninth row, the top and bottom lines of the character are automatically blanked, leaving, effectively, 8 unique lines for display. The 8275 cursor position registers are initialized to zero, forcing the cursor to the upper left-hand corner of the display. The preset counters command resets all 8275 counters to zero and stops the 8275 counters until another command is issued. The 8275 is then started by a start display command. An interrupt will be generated from the 8275 approximately 15 ms later. Interrupts are enabled following the 8275 start command. Interrupts were disabled prior to this time to insure that the central processor did not react to erroneous interrupts from the 8275 generated prior to 8275 initialization. The processor, following initialization, waits in a loop until the arrival of an interrupt from the 8275 or 8251.

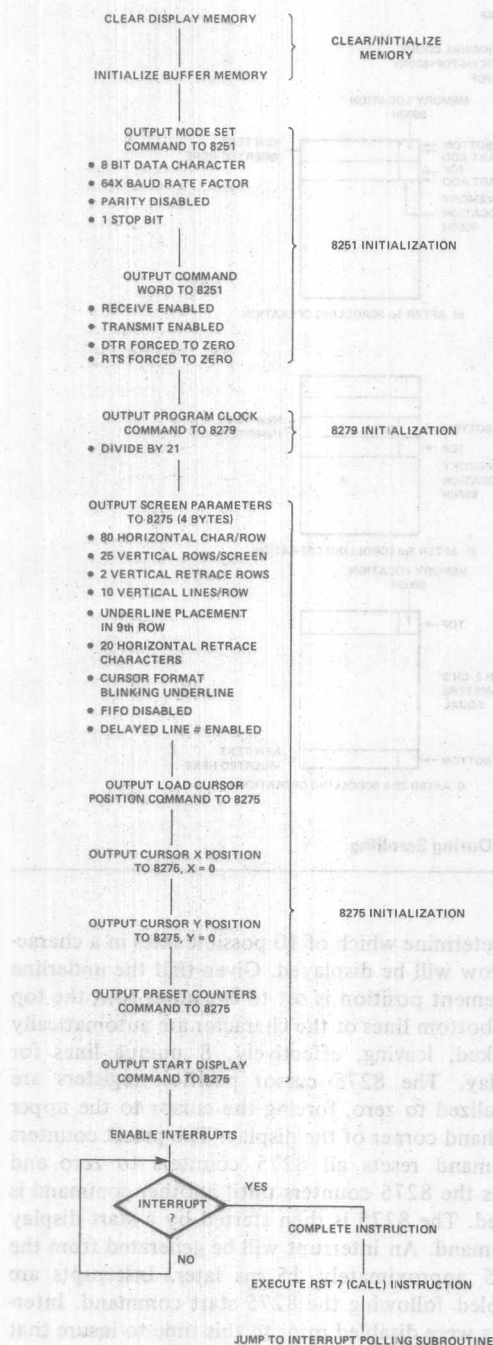


Figure 4-14. System Initialization Routines

### Interrupt Polling Subroutine (Poll)

The interrupt polling subroutine, Figure 4-15, tests to determine the source of the interrupt. If the interrupt originated with the 8275, the 8275 interrupt service subroutine is called. Following completion of the subroutine, interrupts are re-enabled, and a return executed. An interrupt issued from the 8251 forces subroutine calls to the read/store USART character subroutine and the character recognition/handling subroutine. Interrupts are re-enabled at the completion of the character recognition/handling routine. A return operation follows.

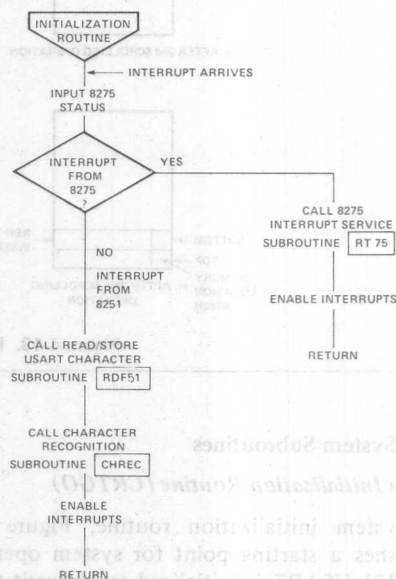


Figure 4-15. Interrupt Polling Subroutine (POLL)

### 8275 Interrupt Service Subroutine (RT 75)

The 8275 interrupt service subroutine, Figure 4-16, re-initializes the 8257 DMA Controller, then tests the 8279 FIFO status. If a character has been transmitted from the keyboard to the Keyboard Controller, a table lookup operation is performed to obtain the correct ASCII code for the character, and the character is transmitted.

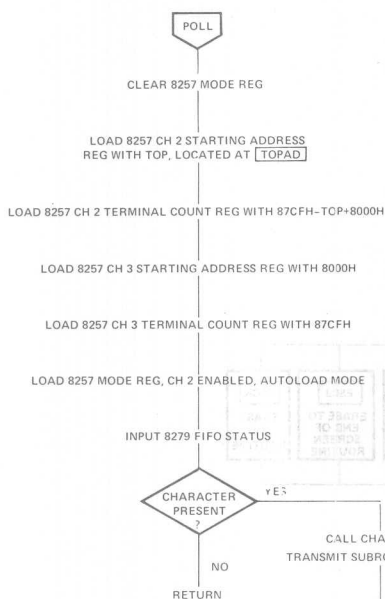


Figure 4-16. 8275 Interrupt Service Subroutine (RT75)

#### USART Read/Store Subroutine (RDF 51)

The read/store USART character subroutine, Figure 4-17, moves a character from the USART to the CPU, masks off the upper-most bit, and stores the character in system buffer memory.

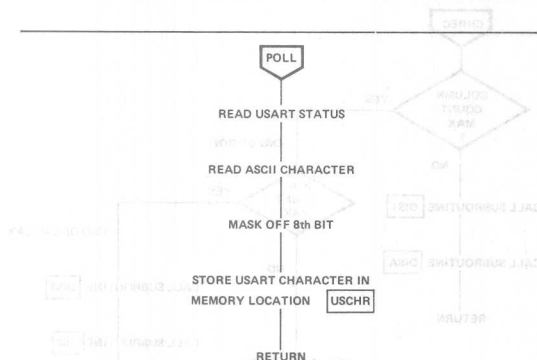


Figure 4-17. READ/STORE USART Character Subroutine (RDF51)

#### Character Recognition/Handling Subroutine (CHREC)

The character recognition/handling subroutine, Figure 4-18, examines the masked USART character to determine whether the character is a displayable character, control code, or the first or second character in an escape sequence. A call to the appropriate subroutine follows the decision-making process. If the character is the first character in an escape sequence, the escape sequence flag is set and the processor loops until a second character is received. The character immediately following the ESC character is examined by the escape code handling subroutine and a jump to an escape code routine follows. If the character is a displayable character or control code, the appropriate subroutine is called.

ter to determine whether the character is a displayable character, control code, or the first or second character in an escape sequence. A call to the appropriate subroutine follows the decision-making process. If the character is the first character in an escape sequence, the escape sequence flag is set and the processor loops until a second character is received. The character immediately following the ESC character is examined by the escape code handling subroutine and a jump to an escape code routine follows. If the character is a displayable character or control code, the appropriate subroutine is called.

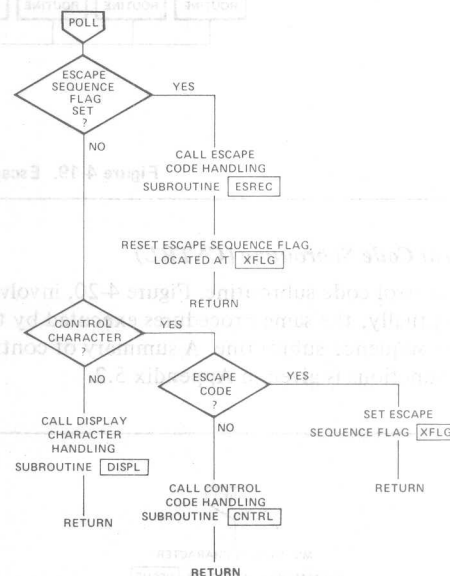


Figure 4-18. Character Recognition/Handling Subroutine (CHREC)

#### Escape Sequence Subroutine (ESREC)

The escape sequence subroutine, Figure 4-19, performs a masking operation on the USART character, shifts the result by one bit position, and adds this value to the base address of the escape sequence lookup table, BSETI. The lookup table contains starting addresses for each of the escape sequence routines. This address is jammed into the program counter and the routine executed. A summary of escape sequence functions is given in Appendix 5.2.

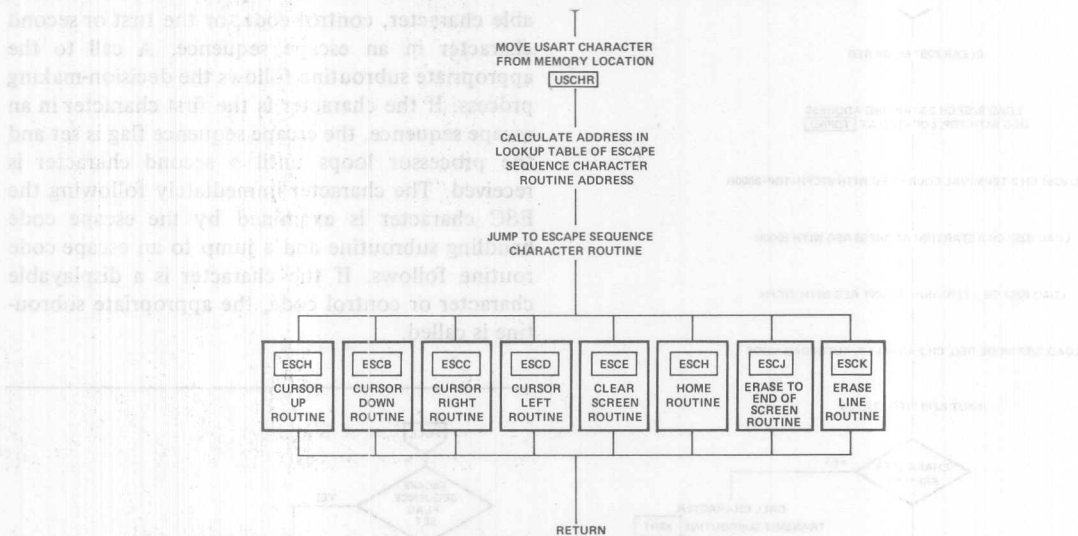


Figure 4-19. Escape Sequence Subroutine (ESREC)

### Control Code Subroutine (CNTRL)

The control code subroutine, Figure 4-20, involves, conceptually, the same procedures executed by the escape sequence subroutine. A summary of control code functions is given in Appendix 5.2.

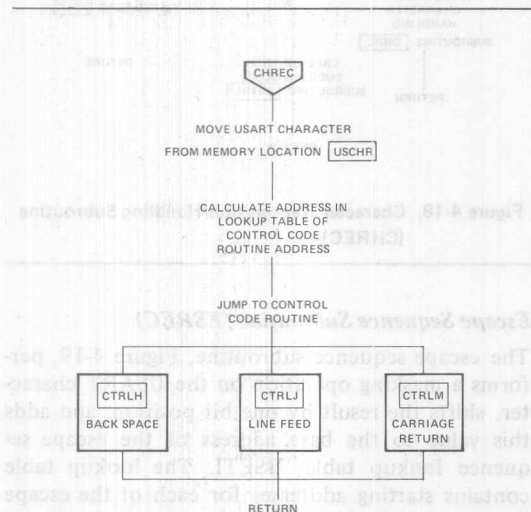


Figure 4-20. Control Code Subroutine (CNTRL)

### Display Character Handling Subroutine (DISPL)

The display character handling subroutine, Figure 4-21, determines if the cursor is located in the last column of the row, the last display position, or elsewhere and calls the appropriate subroutines.

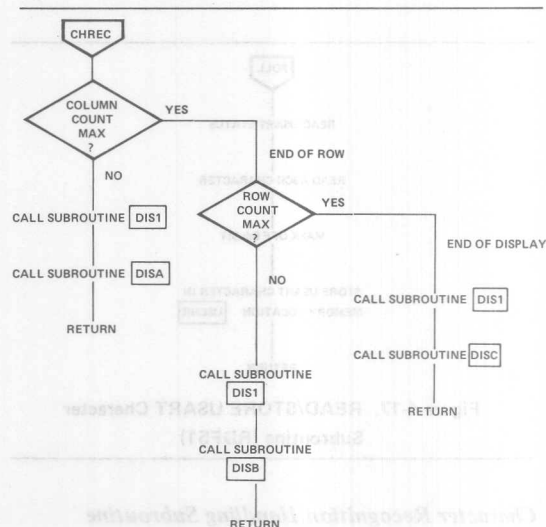


Figure 4-21. Display Character Handling Subroutine (DISPL)

### Display Subroutine One (DIS1)

Display subroutine one, Figure 4-22, calculates the location in memory at which the display character is to be inserted. If the location calculation results in an address outside of the display memory bounds, appropriate compensation action is taken. Prior to inserting the display character in memory, the first character position in the row in which the character will be located is examined. If an End of Row character (EOR) is found, the row in question will be blanked by the 8275. It is necessary to clear the row by filling it with space codes (Fill Subroutine), then insert the display character in the desired location. If no EOR character is found, insertion proceeds without further software intervention.

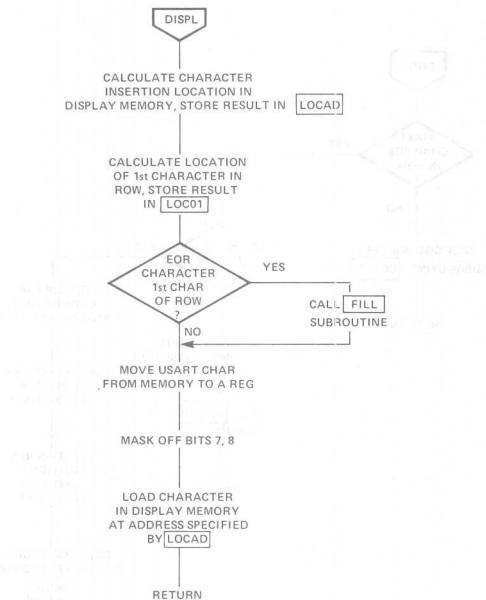
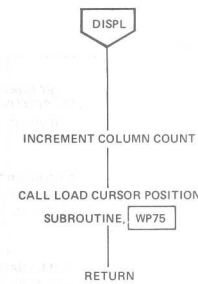


Figure 4-22. Display Subroutine 1 (DIS1)

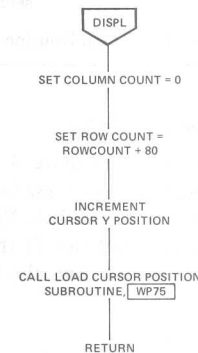
### Display Subroutines A, B, C (DISA, DISB, DISC)

Display subroutines A, B, and C, Figure 4-23, modify the appropriate display memory pointers. The modifications are based on the present cursor location, as determined by subroutine DISPL. The resulting cursor position data is transferred to the 8275 Cursor X and Y Position registers. If DISC is called, a scrolling operation occurs.

#### DISPLAY SUBROUTINE A (DISA)



#### DISPLAY SUBROUTINE B (DISB)



#### DISPLAY SUBROUTINE C (DISC)

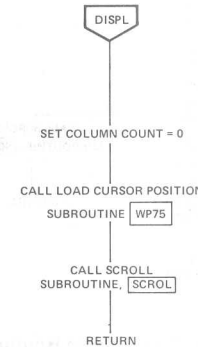


Figure 4-23. Display Subroutines – A (DISA), B (DISB), C (DISC)

### Cursor Up Routine (ESCA)

The cursor up routine, Figure 4-24, determines if the cursor is located in the first display row. If it is, the Row Count and Column Count values are modified, and the cursor is moved to the last display row with no change in X position. If the cursor is not in the top row, the row up subroutine is called.



Figure 4-24. Cursor Up Routine (ESCA)

### Cursor Down Routine (ESCB)

The cursor down routine, Figure 4-25, determines if the cursor is located in the last display row. If it is, the scroll subroutine is called. No modification of cursor position is called for. If the cursor is not located in the last display row, the row down subroutine is called.

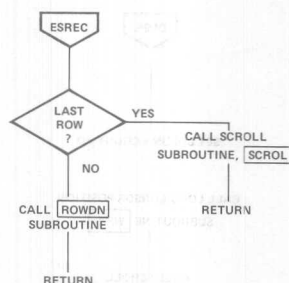


Figure 4-25. Cursor Down Routine (ESCB)

### Cursor Right Routine (ESCC)

The cursor right routine tests the cursor location and moves the cursor as described in Figure 4-26. If the cursor is in the last display position, a scrolling operation occurs. 8275 Cursor X and Y Position registers are updated accordingly.

### Cursor Left Routine (ESCD)

The cursor left routine tests the cursor location and moves the cursor as described in Figure 4-27.

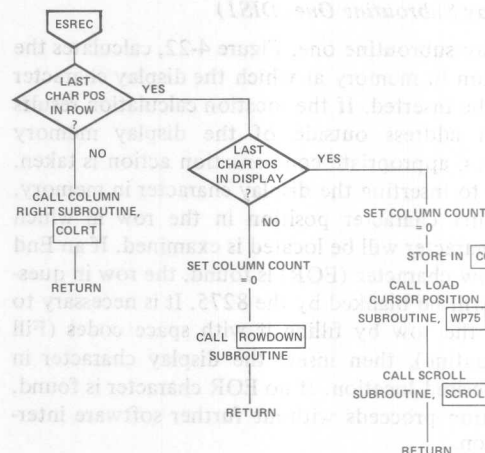


Figure 4-26. Cursor Right Routine (ESCC)

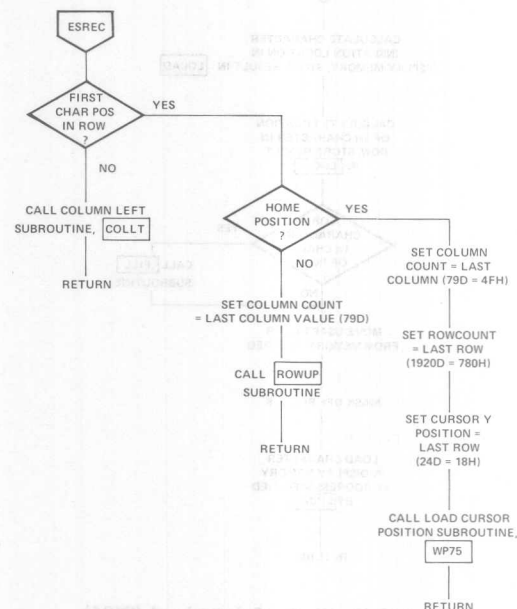


Figure 4-27. Cursor Left Routine (ESCD)

### Clear Screen Routine (ESCE)

Several possibilities existed for implementing the clear screen function. The simplest of these techniques involves filling the display memory with space codes. This technique, although conceptually simple, requires several milliseconds to implement.

The End-of-Row character (EOR) recognized by the 8275 allows the clear screen feature to be executed in a considerably shorter time span. During the clear screen routine, Figure 4-28, EOR characters are placed in the first character position of each row in display memory. Since the EOR character blanks the entire display row when placed in the first character position of the row, the use of EOR characters in each row blanks the entire screen. All pointers are cleared during the clear screen operation.

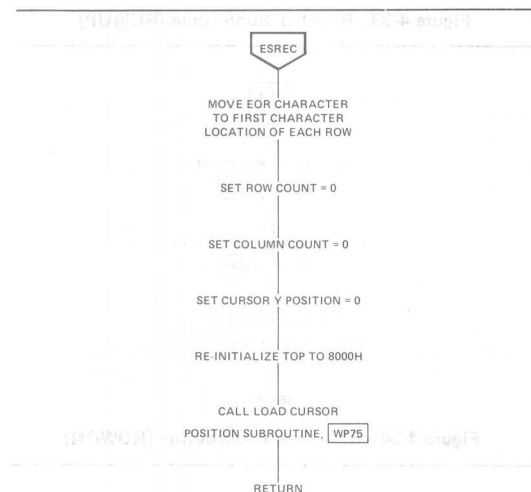


Figure 4-28. Clear Screen Routine (ESCE)

### Home Routine (ESCH)

The home routine, Figure 4-29, resets the Row Count, Column Count and Cursor Y Position buffers to zero, but does not affect the value of TOP.

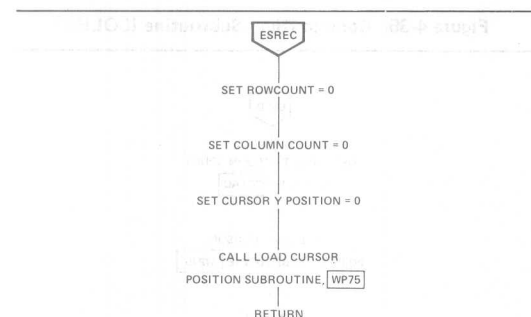


Figure 4-29. Home Routine (ESCH)

### Erase to End of Screen Routine (ESCJ)

The erase to end of screen routine, Figure 4-30, inserts End of Row characters (EOR) in display memory in the same fashion as the clear screen routine. The fundamental difference between the routines is that the erase to end of screen routine must insert EOR characters selectively. Only rows from the present display row until the last display row, pointed to by BOTTOM, receive EOR characters. It should be noted that the pointer BOTTOM changes dynamically with scrolling operations.

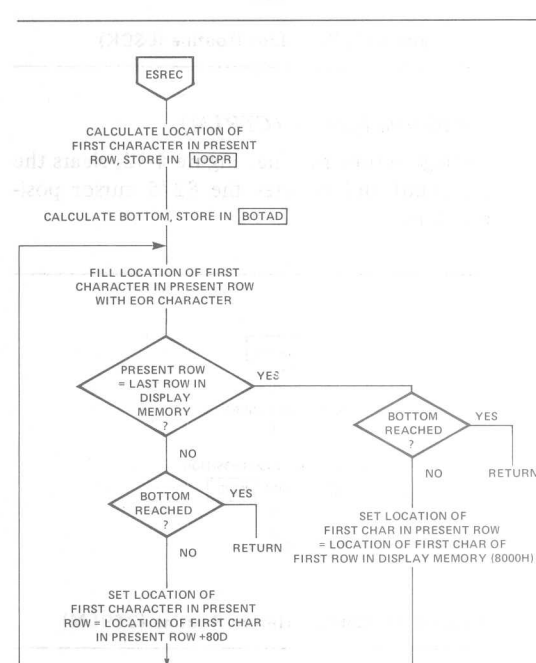


Figure 4-30. Erase to End of Screen Routine (ESCJ)

### Erase Line Routine (ESCK)

The erase line routine, Figure 4-31, calculates the location of the first character in the current display row, stores the location in buffer memory, and calls the fill subroutine, which fills the row with space codes.

### Backspace Routine (CTRLH)

See cursor left routine.

### Line Feed Routine (CTRLJ)

See cursor down routine.

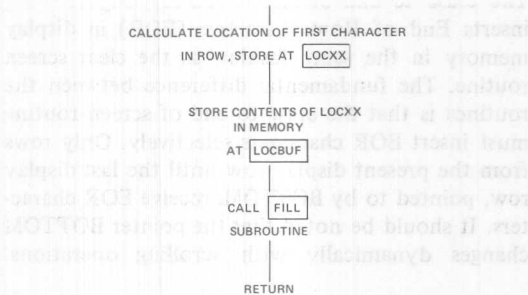


Figure 4-31. Erase Line Routine (ESCK)

### Carriage Return Routine (CTRLM)

The carriage return routine, Figure 4-32, clears the column count and updates the 8275 cursor position registers.

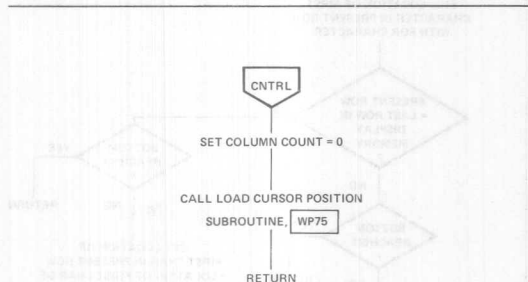


Figure 4-32. Carriage Return Routine (CTRLM)

### Row Up, Row Down Subroutines (ROW UP, ROW DOWN)

The row up subroutine, Figure 4-33, subtracts 80D from the Row Count value, decrements the Cursor Y Position pointer, and updates the 8275 Cursor Position registers. The row down subroutine, Figure 4-34, differs in that 80D is added to Row Count.

### Column Right, Column Left Subroutines (COLRT, COLLT)

The column right subroutine, Figure 4-35, increments the Column Count pointer and updates the 8275 cursor position registers. The column left subroutine, Figure 4-36, differs in that the Column Count is decremented.

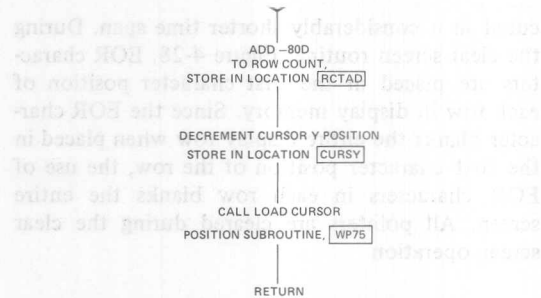


Figure 4-33. Row Up Subroutine (ROWUP)

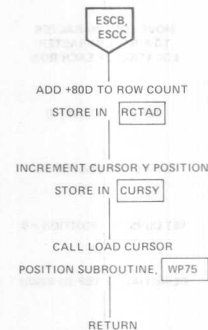


Figure 4-34. Row Down Subroutine (ROWDN)

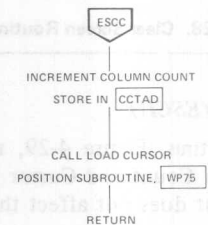


Figure 4-35. Column Right Subroutine (COLRT)

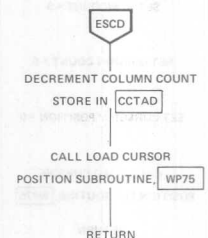


Figure 4-36. Column Left Subroutine (COLLT)

### Scroll Subroutine (SCROL)

The scroll subroutine, Figure 4-37, fills the row in display memory pointed to by TOP with space characters via the fill subroutine, then modifies the value of TOP. TOP is utilized by the 8275 service subroutine in re-initializing the 8257 DMA controller.

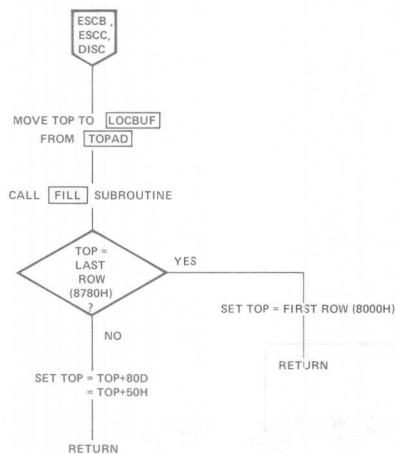


Figure 4-37. Scroll Subroutine (SCROL)

### Fill Subroutine (FILL)

The fill subroutine, Figure 4-38, calculates the location of the last character in the current display row, plus one character position, by adding 80D = 50H to the location of the first character in the current display row. The current stack pointer value is saved, then the stack pointer is loaded with the location of the last character in the current display row, plus one character position. The B and C registers of the CPU are loaded with space characters and 40 PUSH B operations performed. This technique provides a rapid means (275  $\mu$ sec) of filling a given row with space codes.

### Load Cursor Position Subroutine (WP 75)

The load cursor position subroutine, Figure 4-39, transfers the contents of the Column Count and cursor Y position pointers to the 8275 cursor X position and cursor Y position registers, respectively.

The relationship between system subroutines is presented in Appendix 5.3. Software timing considerations are covered in Appendix 5.4.

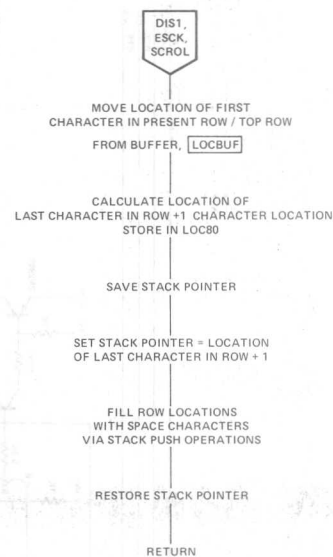


Figure 4-38. Fill Subroutine (FILL)

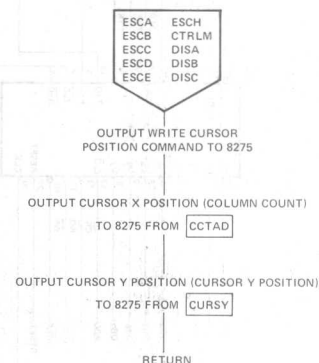
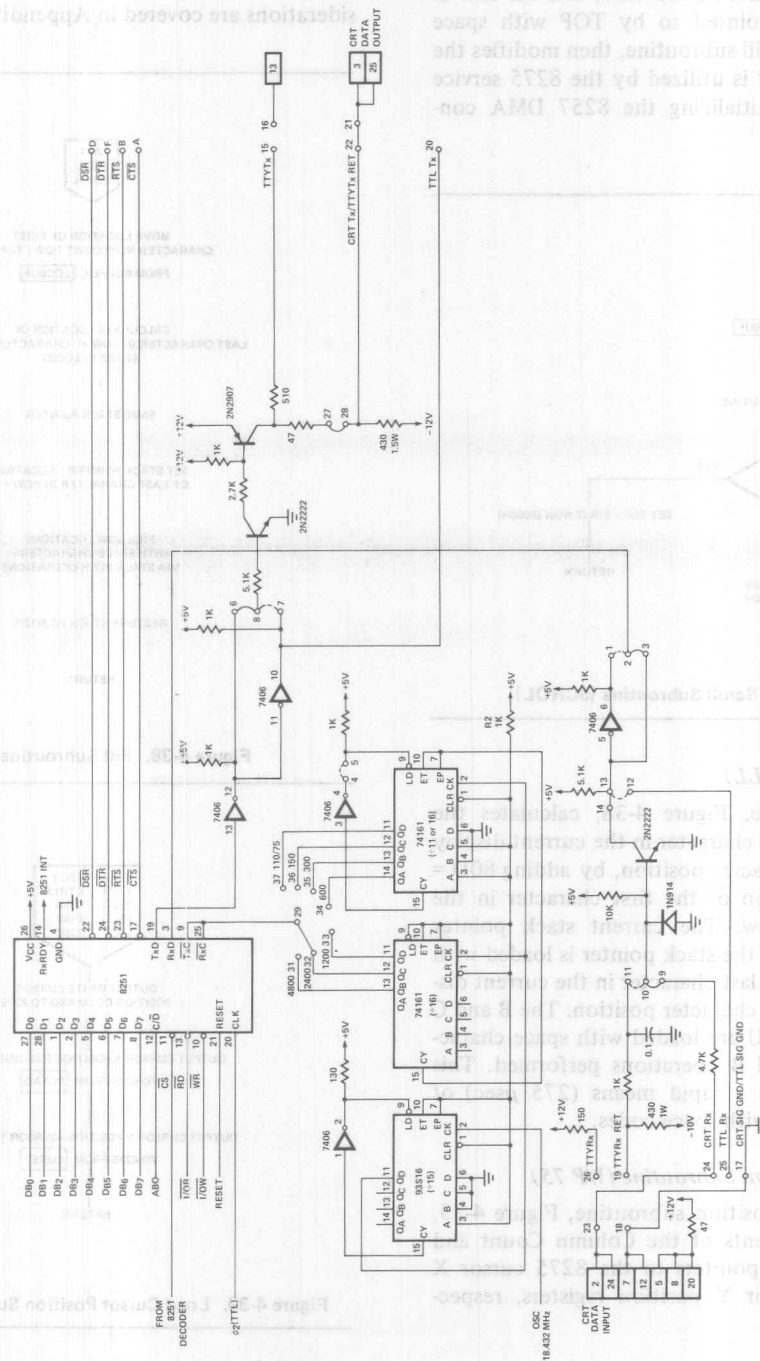
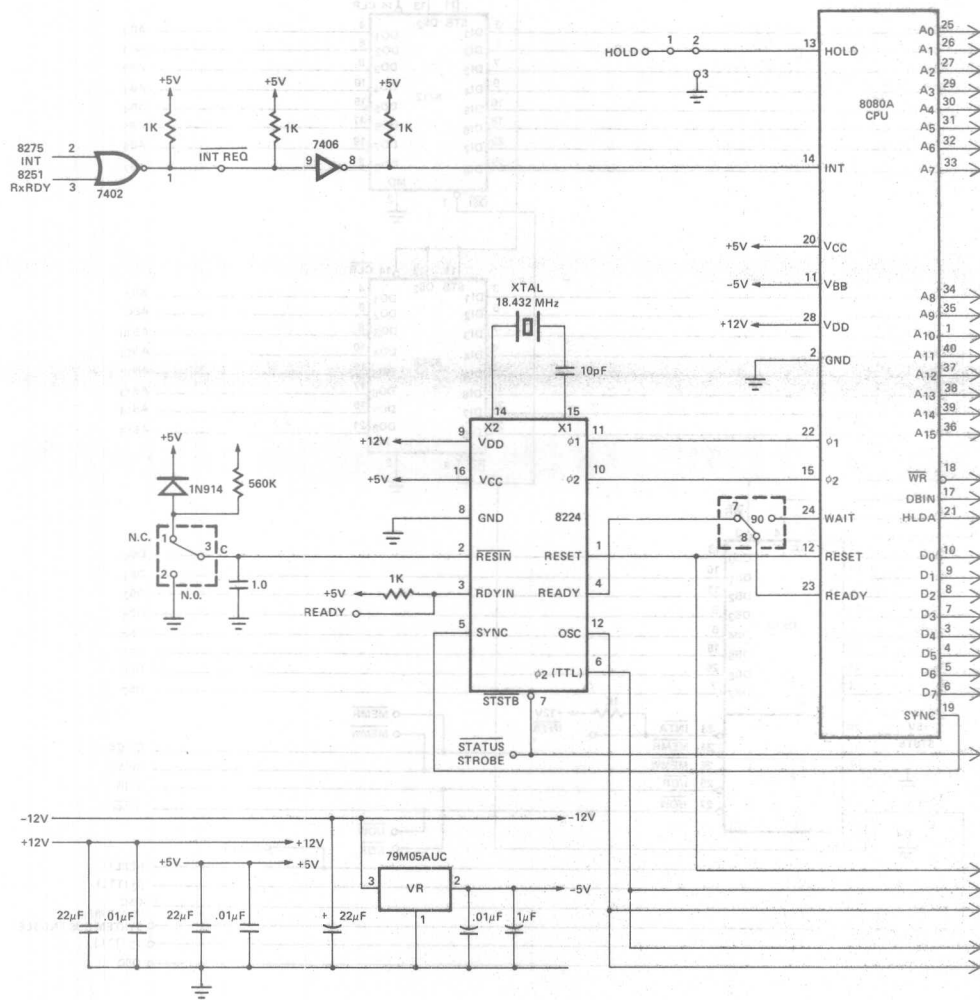


Figure 4-39. Load Cursor Position Subroutine (WP75)

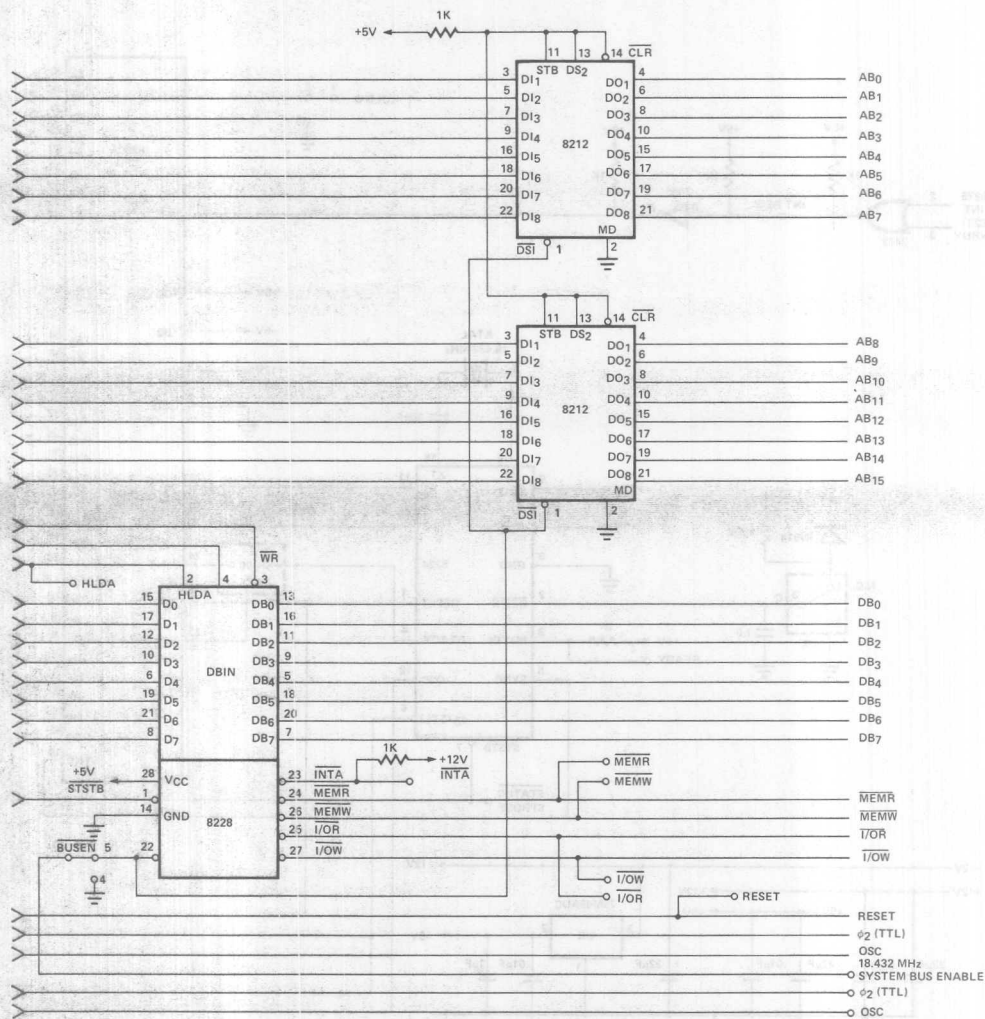
# SERIAL COMMUNICATIONS SECTION

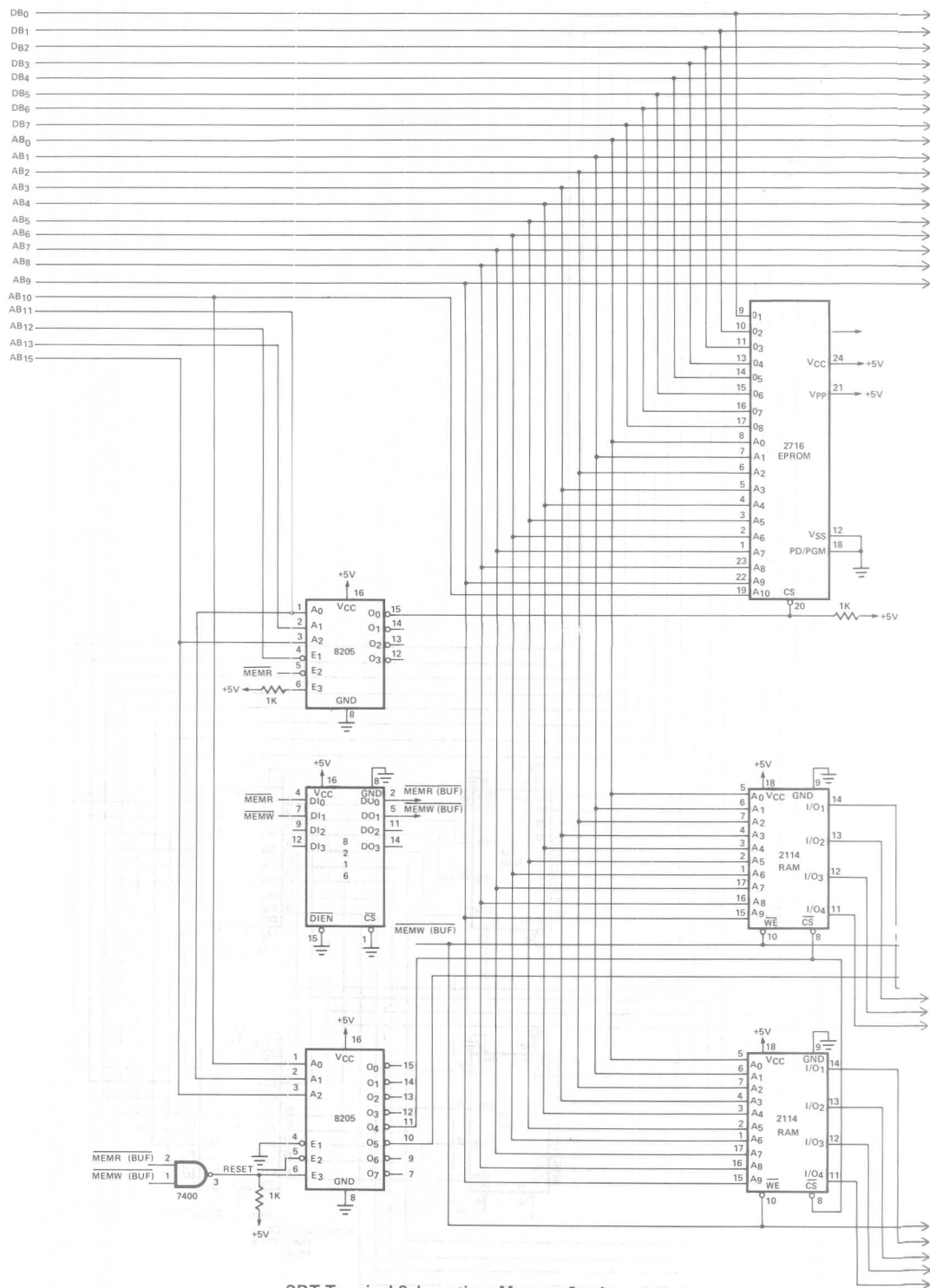


## Appendix 5.1

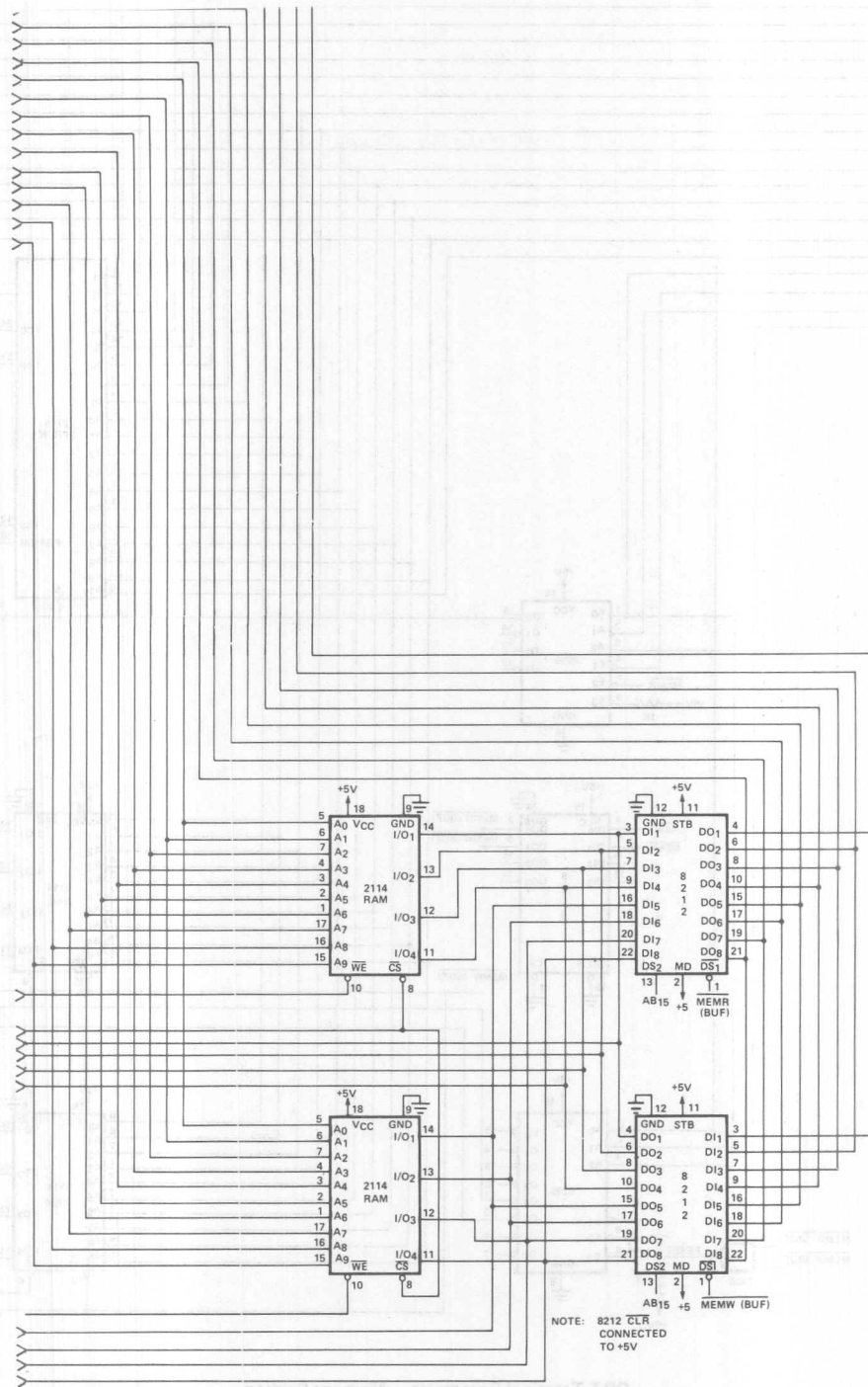


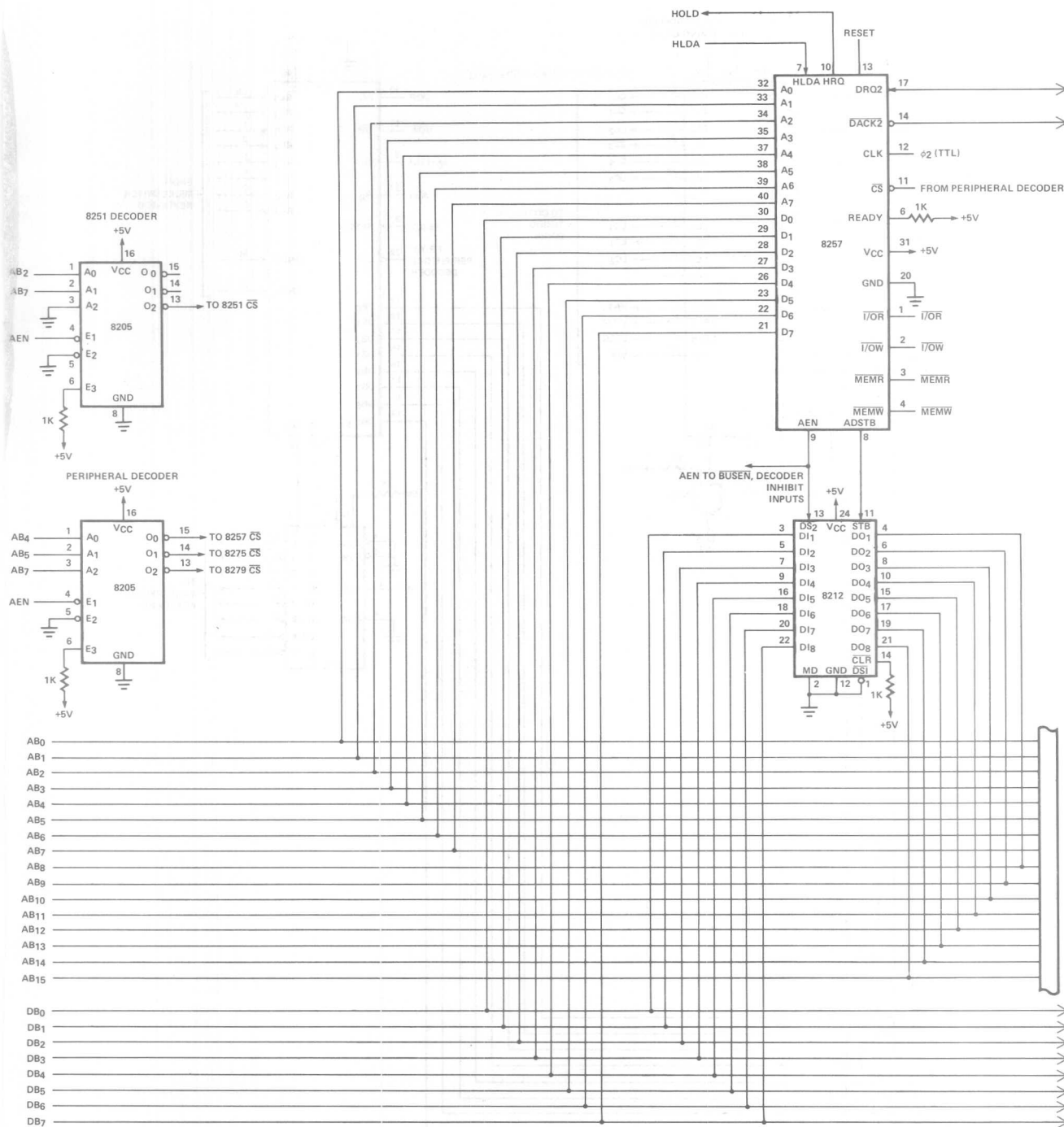
CRT Terminal Schematic – CPU Section



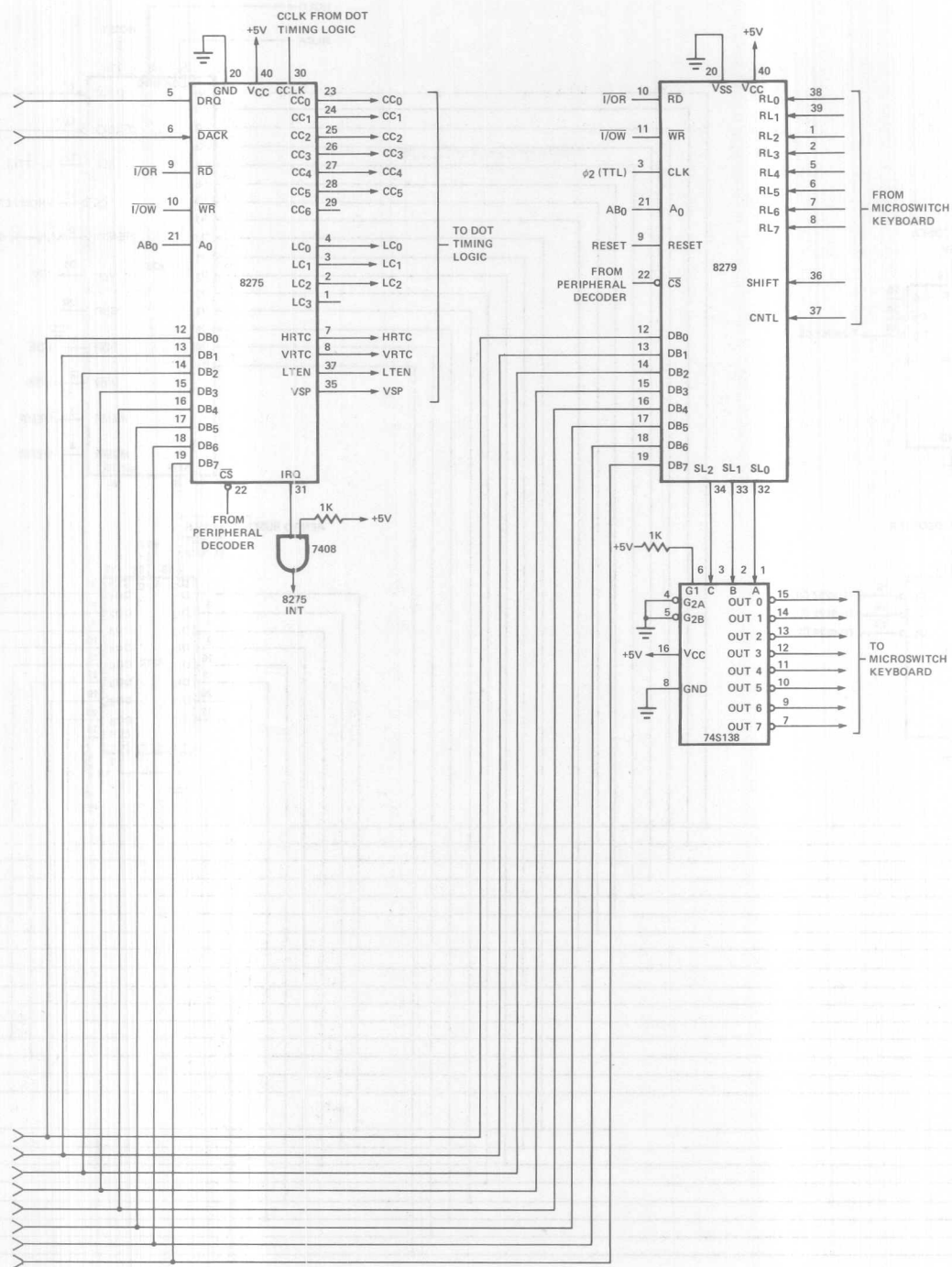


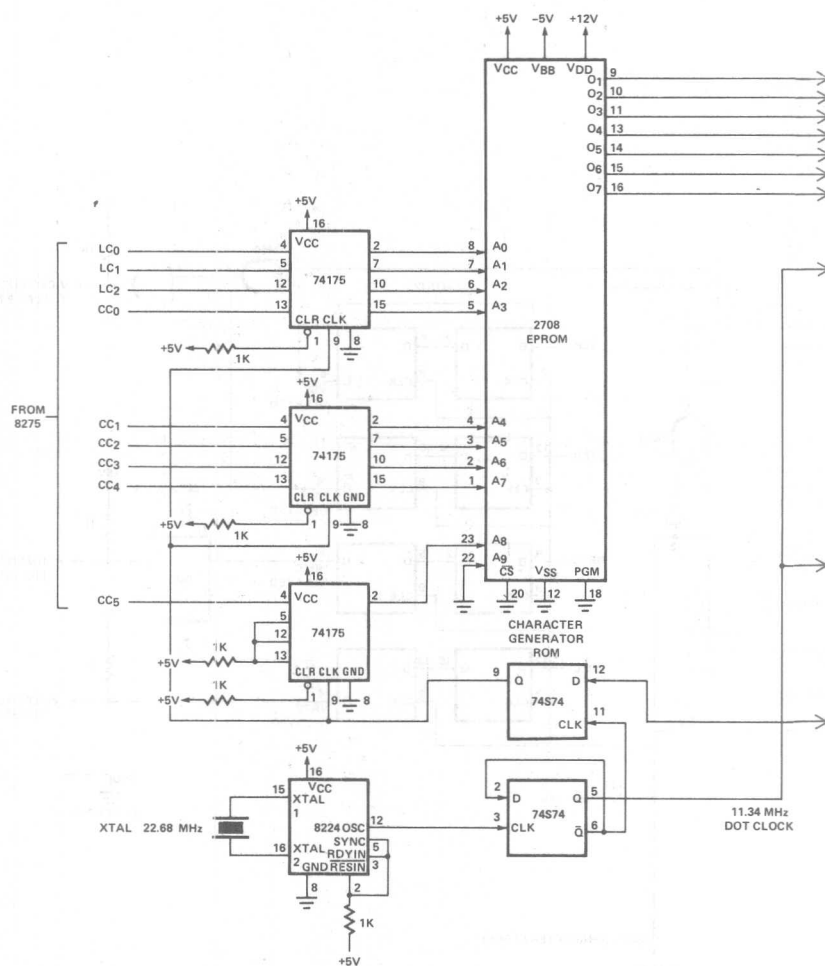
CRT Terminal Schematic – Memory Section





CRT Terminal Schematic — Peripherals Section





CRT Terminal Schematic — Dot Timing Logic Section

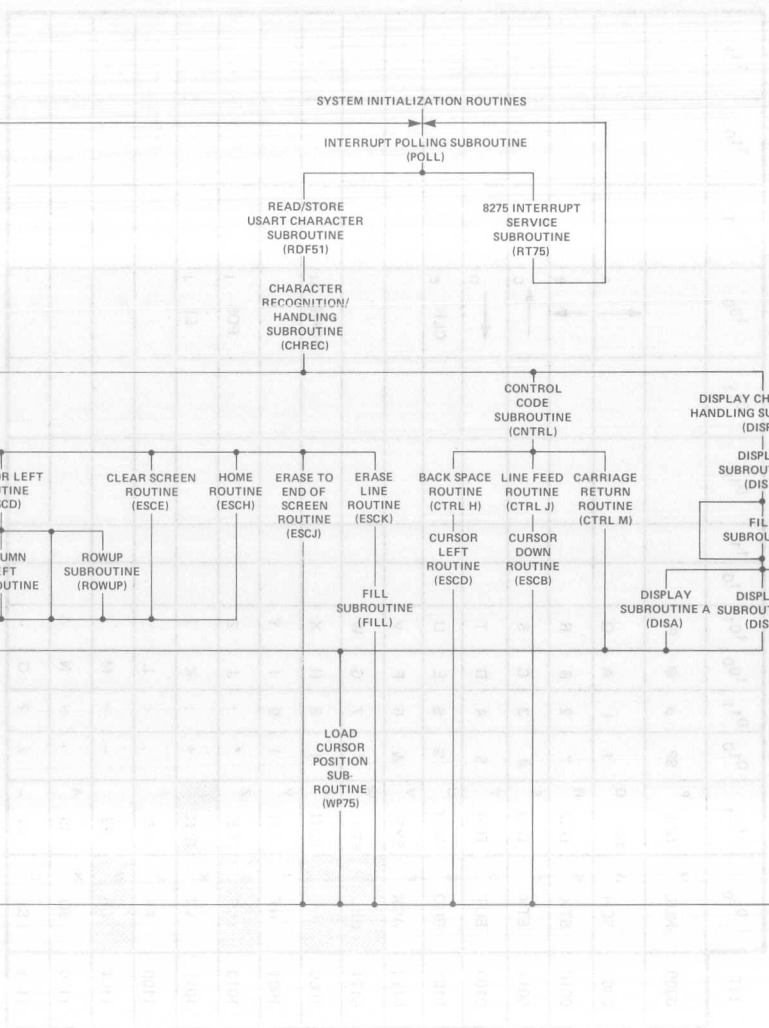


## Appendix 5.2 ESCAPE/CONTROL/DISPLAY CHARACTER SUMMARY

BIT	CONTROL CHARACTERS				DISPLAYABLE CHARACTER				ESCAPE SEQUENCE					
	0 <sub>0</sub> 0	0 <sub>0</sub> 1	0 <sub>1</sub> 0	0 <sub>1</sub> 1	1 <sub>0</sub> 0	1 <sub>0</sub> 1	1 <sub>1</sub> 0	1 <sub>1</sub> 1	0 <sub>1</sub> 0	0 <sub>1</sub> 1	1 <sub>0</sub> 0	1 <sub>0</sub> 1	1 <sub>1</sub> 0	1 <sub>1</sub> 1
0000	NUL @	DLE P	SP ϕ	@ P										
0001	SOH A	DC1 Q	! I	A Q							↑ A			
0010	STX B	DC2 R	" 2	B R							↓ B			
0011	ETX C	DC3 S	# 3	C S							→ C			
0100	EOT D	DC4 T	\$ 4	D T							← D			
0101	ENQ E	NAK U	% 5	E U							CLR E			
0110	ACK F	SYN V	& 6	F V										
0111	BEL G	ETB W	' 7	G W										
1000	BS H	CAN X	( 8	H X							HOME H			
1001	HT I	EM Y	) 9	I Y										
1010	LF J	SUB Z	* :	J Z							EOS I			
1011	VT K	ESC	+ :	K [							EL J			
1100	FF L	FS /	, <	L \										
1101	CR M	GS	- =	M ]										
1110	SO N	RS ^	· >	N ^										
1111	SI O	US -	/ ?	O -										

NOTE: Shaded blocks = functions terminal will react to. Others can be generated but are ignored up on receipt.

### Appendix 5.3 E INTERREL



## Appendix 5.4 SOFTWARE TIMING

Subroutine execution times are summarized in the flowchart provided in Figure 5-1. The values shown represent the number of clock cycles required for the execution of a given routine. The actual routine execution time is obtained by multiplying the number of clock cycles/routine by the time/clock cycle. For a 2.048 MHz system clock, the time/clock cycle is 0.4883  $\mu$ sec. It should be noted that the values indicated represent worst-case execution times. In order to appreciate the meaning of the subroutine execution times, it is necessary to consider two factors:

1. The time available for the CPU to execute instructions between DMA operations.
2. The maximum rate at which data characters are presented to the CPU for processing.

CPU availability during a complete display frame is illustrated in Figure 5-2. Available CPU processing time, per character, at 4800 baud, during the DMA active portion of the display frame, is illustrated in Figure 5-3. It can be seen from Figure 5-3 that 1443  $\mu$ sec are available for processing each character during the DMA active portion of the frame. Total CPU processing time during the DMA inactive portion of the frame may be seen from Figure 5-2 to be 1234  $\mu$ sec. This value encompasses the time to process the 8275 interrupt and perform character handling functions.

Using the information contained in Figure 5-1, the maximum execution time\* for a given character handling routine is 802  $\mu$ sec. Since this value is less than 1.443 msec, proper timing is assured. Using the maximum character handling routine execution time and the time required for 8275 interrupt processing, the maximum CPU availability requirement during the DMA inactive portion of the frame may be calculated. This value corresponds to 802  $\mu$ sec + 253  $\mu$ sec (8275 interrupt processing) or 1055  $\mu$ sec. Since this value is less than 1234  $\mu$ sec, proper timing is assured.

\*see notes, Figure 5-1.

## Appendix 5.5 VISUAL ATTRIBUTE IMPLEMENTATION CONSIDERATIONS

In order to utilize the visual attribute features of the 8275, it is necessary to modify the CRT system hardware and software functions accordingly.

Hardware modifications necessary to implement character attributes are illustrated in Figure 5-4. The attribute outputs LA0-LA1 selectively control the data transferred to the output shift register.

The software memory management scheme presented in the Application Note must be modified in order to accommodate attribute features. An outline of the software considerations involved when using the attribute features is presented as follows:

1. Attributes, as described in the 8275 Data Sheet, occupy character locations in display memory. Since the number of attributes per display row may be variable, the linear mapping relationship between character position on the screen and memory pointers Top, Row Count, and Column Count no longer exists. It is necessary to keep track of the number of attribute characters in each row and their specific location when modifying pointer values.
2. The increased number of character locations required will force the user to incorporate additional display RAM.
3. Since the total number of characters in display memory may be variable when attributes are utilized, it is necessary to modify the starting address and terminal count values for the DMA channels as required.
4. Character insertion and deletion operations may be handled through block transfer operations or through the use of extended display memory row segments.

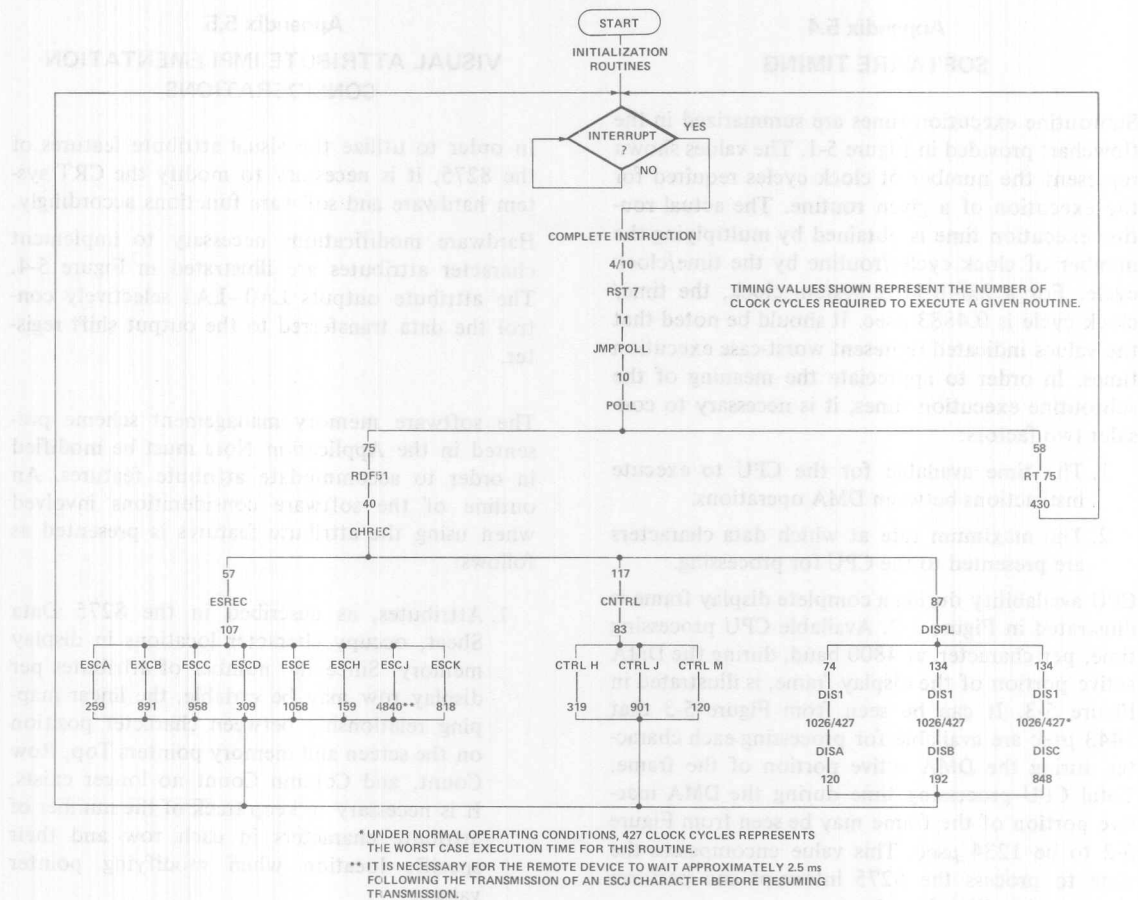


Figure 5-1. Subroutine Execution Times Flowchart

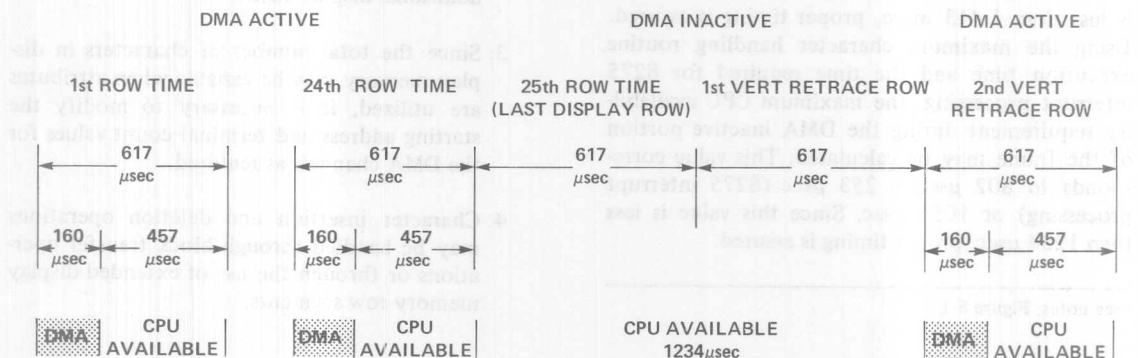
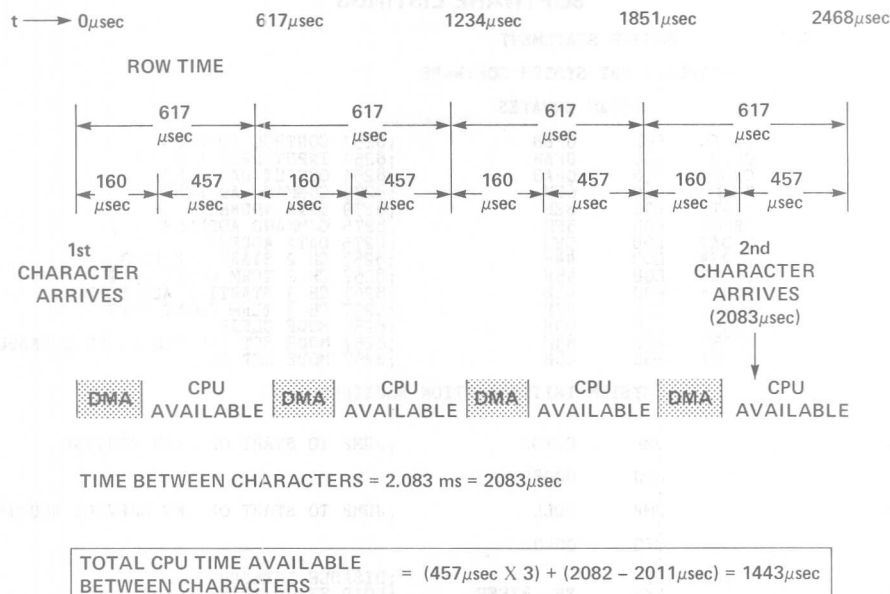


Figure 5-2. CPU Availability



BAUD RATE = 4800 BAUD  
 10 BITS/CHARACTER

Figure 5-3. CPU Availability/Character at 4800 Baud (DMA Active)

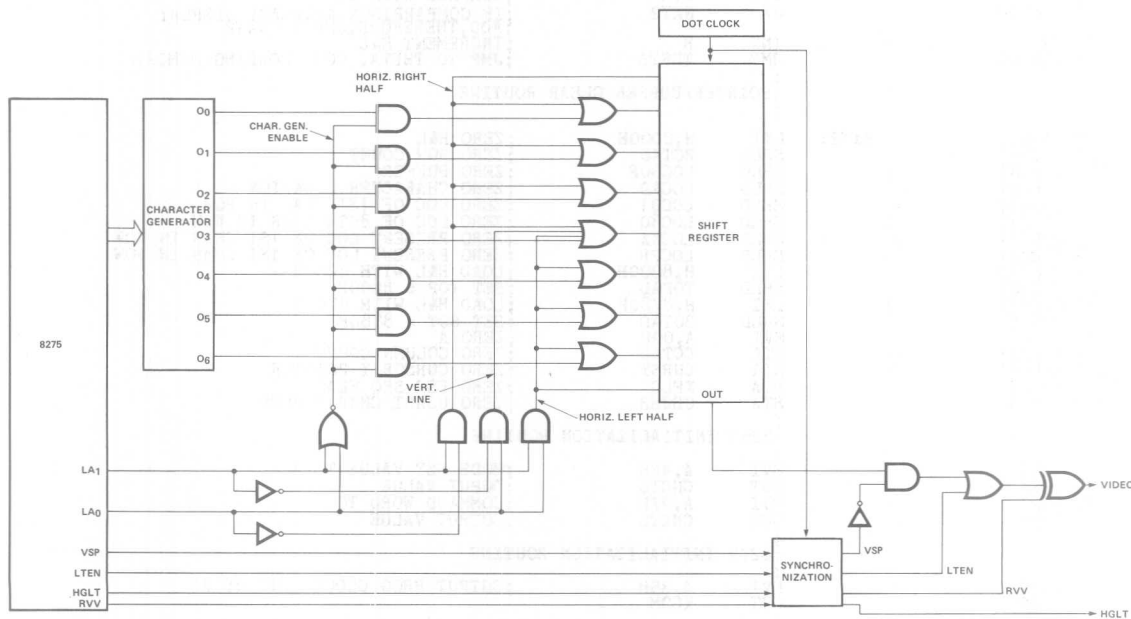


Figure 5-4. Typical Character Attribute Logic

LOC	OBJ	SEQ	SOURCE STATEMENT
		1	;8275/8279 CRT SYSTEM SOFTWARE
		2	;
		3	;
		4	;
		5	;
00FB		5	CNCTL EQU 0FBH ;8251 CONTROL ADDRESS
00FA		6	CNIN EQU 0FAH ;8251 INPUT DATA ADD
00FA		7	CNOUT EQU 0FAH ;8251 OUTPUT DATA ADD
006F		8	KCOM EQU 6FH ;8279 COMMAND ADDRESS
006E		9	KDAT EQU 6EH ;8279 DATA ADDRESS
005F		10	CRCOM EQU 5FH ;8275 COMMAND ADDRESS
005E		11	CRDAT EQU 5EH ;8275 DATA ADDRESS
0044		12	PC2SA EQU 44H ;8257 CH 2 START ADD PORT
0045		13	PC2TC EQU 45H ;8257 CH 2 TERM COUNT PORT
0046		14	PC3SA EQU 46H ;8257 CH 3 STARTING ADD PORT
0047		15	PC3TC EQU 47H ;8257 CH 3 TERM COUNT PORT
0000		16	MDC57 EQU 00H ;8257 MODE CLEAR
0084		17	MDS57 EQU 84H ;8257 MODE SET (AUTOLOAD, CH 2 ENABLED)
0048		18	PMD57 EQU 48H ;8257 MODE SET PORT
		19	;
		20	;
		21	;
		22	;
0000	C34000	23	JMP CRTGO ;JUMP TO START OF MAIN ROUTINE
0038		24	ORG 0038H
		25	;
0038	C3C900	26	JMP POLL ;JUMP TO START OF INT SERVICE ROUTINE
		27	;
0040		28	ORG 0040H
		29	;
0040	F3	30	CRTGO: DI ;DISABLE INTERRUPTS
0041	31FF87	31	LXI SP, 87FFH ;LOAD STACK POINTER
		32	;
		33	;
		34	;
		35	;
		36	;
0044	210080	37	LXI H, 8000H ;LOAD H&L WITH START ADD OF DISPLAY MEM
0047	3E20	38	MVI A, 20H ;LOAD A WITH SPACE CHAR CODE
0049	77	39	MOV M, A ;LOAD SPACE CHAR IN MEM
004A	7D	40	MOV A, L ;MOVE LOW ADD BYTE TO A
004B	FECD	41	CPI 0CFH ;COMPARE WITH 0CFH
004D	CA5400	42	JZ NXT1 ;IF COMPARRISON JMP TO NXT1
0050	23	43	INX H ;INCREMENT H&L
0051	C34700	44	JMP THETA ;JMP TO THETA, CONT LOADING MEMORY
0054	7C	45	MOV A, H ;MOVE UP ADD BYTE TO A
0055	FE87	46	CPI 87H ;COMPARE WITH 87H
0057	CA5E00	47	JZ NXT2 ;IF COMPARRISON, ADD LAST DISPLAY
		48	ADD, THEREFORE, JMP TO NXT2
005A	23	49	INX H ;INCREMENT H&L
005B	C34700	50	JMP THETA ;JMP TO THETA, CONT LOADING MEMORY
		51	;
		52	;
		53	;
		54	;
		55	;
		56	;
		57	;
		58	;
		59	;
		60	;
		61	;
		62	;
		63	;
		64	;
		65	;
		66	;
		67	;
		68	;
		69	;
		70	;
		71	;
		72	;
		73	;
		74	;
		75	;
		76	;
		77	;
		78	;
		79	;
		80	;
		81	;
		82	;
		83	;
		84	;

009C	3E00	85	MVI	A,00H	;RESET AND STOP DISPLAY
009E	D35F	86	OUT	CRCOM	
00A0	3E4F	87	MVI	A,4FH	;SCREEN PARAM BYTE 1
00A2	D35E	88	OUT	CRDAT	
00A4	3E58	89	MVI	A,58H	;BYTE 2
00A6	D35E	90	OUT	CRDAT	
00A8	3E89	91	MVI	A,89H	;BYTE 3
00AA	D35E	92	OUT	CRDAT	
00AC	3ED9	93	MVI	A,0D9H	;BYTE 4
00AE	D35E	94	OUT	CRDAT	
00B0	3E80	95	MVI	A,80H	;LOAD CURSOR POSITION
00B2	D35F	96	OUT	CRCOM	
00B4	3E00	97	MVI	A,00H	;CURSOR X POSITION
00B6	D35E	98	OUT	CRDAT	
00B8	3E00	99	MVI	A,00H	;CURSOR Y POSITION
00BA	D35E	100	OUT	CRDAT	
00BC	3EE0	101	MVI	A,0E0H	;PRESET COUNTERS
00BE	D35F	102	OUT	CRCOM	
00C0	3E23	103	MVI	A,23H	;START DISPLAY
00C2	D35F	104	OUT	CRCOM	
00C4	FB	105	EI		;ENABLE INTERRUPTS
00C5	00	106	LOOP: NOP		
00C6	C3C500	107	JMP	LOOP	
		108			
		109			
		110			;8275/8251 INTERRUPT POLLING ROUTINE
		111			
		112			
00C9	DB5F	113	POLL: IN	CRCOM	;READ 8275 STATUS, CLEARING INT
00CB	E620	114	ANI	20H	;MASK STATUS, SAVE INT REQ BIT
00CD	CAD500	115	JZ	AGGIE	;IF STATUS=1, SERVICE 8275
		116			
00D0	CD7304	117	GIGEM: CALL	RT75	;CALL 8275 INT SERVICE SUBROUTINE
00D3	FB	118	EI		;ENABLE INTERRUPTS
00D4	C9	119	RET		;RETURN
		120			
00D5	CDD00	121	AGGIE: CALL	RDF51	;CALL READ USART CHAR ROUTINE
00D8	CDE500	122	CALL	CHREC	;CALL CHARACTER RECOG/HANDLING ROUTINE
00DB	FB	123	EI		;ENABLE INTERRUPTS
00DC	C9	124	RET		;RETURN
		125			
		126			;USART READ/STORE CHAR SUBROUTINE
		127			
00DD	DBFA	128	RDF51: IN	CNIN	;READ ASCII CHAR FROM USART, RESETTNG RXRDY
00DF	E67F	129	ANI	7FH	;MASK BIT 8,SAVE BITS 1-7
00E1	32E587	130	STA	USCHR	;STORE USART CHAR IN MEMORY
00E4	C9	131	RET		;RETURN
		132			
		133			;CHARACTER RECOGNITION/HANDLING SUBROUTINE
		134			
00E5	3AE487	135	CHREC: LDA	XFLG	;LOAD A WITH ESC SEQ FLAG
00E8	E6FF	136	ANI	OFFH	;SET/RESET ZERO BIT
00EA	CAF100	137	JZ	NXTX	;IF ONE,CHAR=2ND CHAR IN ESC SEQ
00ED	CD0F01	138	CALL	ESREC	;CALL ESC SEQ SUBROUTINE
00F0	C9	139	RET		;RETURN
00F1	3AE587	140	NXTX: LDA	USCHR	;LOAD USART CHAR IN A
00F4	E660	141	ANI	60H	;MASK BITS 1-5,&8,SAVING BITS 6&7
00F6	CAF00	142	JZ	NXTY	;IF ZERO CHAR=CONTROL CHAR
		143			;IF ONE CHAR=DISPLAY CHAR
00F9	CD4B03	144	CALL	DISPL	;CALL DISPLAY CHAR SUBROUTINE
00FC	C9	145	RET		;RETURN
00FD	3AE587	146	NXTY: LDA	USCHR	;LOAD USART CHAR IN A
0100	E610	147	ANI	10H	;MASK OFF BITS,SAVE BIT 5
0102	C20901	148	JNZ	NXTZ	;IF ZERO CONT CHAR=CONT CODE
		149			;IF ONE CONT CHAR=ESC CODE
0105	CD2701	150	CALL	CNTRL	;CALL CONTROL CODE SUBROUTINE
0108	C9	151	RET		;RETURN
0109	21E487	152	NXTZ: LXI	H,XFLG	;LOAD H&L WITH ADD OF ESC SEQ FLAG
010C	3601	153	MVI	M,01H	;SET ESC SEQ FLAG
010E	C9	154	RET		;RETURN
		155			
		156			;ESCAPE SEQUENCE SUBROUTINE
		157			
010F	3E00	158	ESREC: MVI	A,00H	;ZERO A
0111	32E487	159	STA	XFLG	;RESET ESC SEQ FLAG
0114	3AE587	160	LDA	USCHR	;LOAD USART CHAR IN A
0117	E60F	161	ANI	0FH	;MASK BITS 5-8
0119	07	162	RLC		;SHIFT LEFT,YIELDING OFFSET
011A	21D004	163	LXI	H,BSET1	;LOAD BASE ADD OF TABLE 1 IN H&L
011D	110000	164	LXI	D,0000H	;ZERO D&E
0120	5F	165	MOV	E,A	;LOAD OFFSET IN E
0121	19	166	DAD	D	;ADD OFFSET TO BASE, RESULT IN H&L
0122	5E	167	MOV	E,M	;MOVE LOW BYTE OF ROUTINE ADD TO E
0123	23	168	INX	H	;INCREMENT COMPUTED ADDRESS
0124	56	169	MOV	D,M	;MOVE UP BYTE OF ROUTINE ADD TO D
0125	EB	170	XCHG		;EXCHANGE D&E WITH H&L
0126	E9	171	PCHL		;LOAD PC WITH ROUTINE ADD, JMP TO ROUTINE
		172			
		173			;CONTROL CODE SUBROUTINE

0127 3AE587	174				
012A E606	175	CNTRL:	LDA	USCHR	;LOAD USART CHAR IN A
012C 21F004	176		ANI	06H	;MASK CHAR, SAVE BITS 2-3
012F 110000	177		LXI	H,BSET2	;LOAD BASE ADD OF TABLE 2 IN H&L
0132 5F	178		LXI	D,0000H	;CLEAR D&E
0133 19	179		MOV	E,A	;LOAD OFFSET IN E
0134 5E	180		DAD	D	;ADD OFFSET TO BASE, RESULT IN H&L
0135 23	181		MOV	E,M	;MOVE LOW BYTE OF ROUTINE ADD TO E
0136 56	182		INX	H	;INCREMENT COMPUTED ADDRESS
0137 EB	183		MOV	D,M	;MOVE UP BYTE OF ROUTINE ADD TO D
0138 E9	184		XCHG		;EXCHANGE D&E WITH H&L
	185		PCHL		;LOAD PC WITH ROUTINE ADD, JMP TO ROUTINE
	186				
	187				
	188				
0139 2AD387	189	ESCA:	LHLD	RCTAD	;LOAD ROWCOUNT IN H&L
013C 7D	190		MOV	A,L	;MOVE LOW BYTE OF ROWCOUNT TO A
013D FE00	191		CPI	00H	;COMPARE BYTE WITH 00H
013F CA4601	192		JZ	ALPHA	;IF BYTE=0 CONTINUE COMPARRISON
0142 CD0803	193		CALL	ROWUP	;CALL ROWUP SUBROUTINE
0145 C9	194		RET		;RETURN
0146 7C	195	ALPHA:	MOV	A,H	;MOVE UP BYTE OF ROWCOUNT TO A
0147 FE00	196		CPI	00H	;COMPARE BYTE WITH 00H
0149 CA5001	197		JZ	BETA	;IF BYTE=0,ROWCOUNT=FIRST ROW
014C CD0803	198		CALL	ROWUP	;CALL ROWUP SUBROUTINE
014F C9	199		RET		;RETURN
0150 218007	200	BETA:	LXI	H,0780H	;LOAD H&L WITH ROWCOUNT=LAST ROW VALUE (1920D)
0153 22D387	201		SHLD	RCTAD	;STORE 0780H IN ROWCOUNT BUFFER
0156 3E18	202		MVI	A,18H	;LOAD A WITH CURSOR Y POS=LAST ROW VALUE (240)
0158 32D587	203		STA	CURSY	;STORE 18H IN CURSOR Y POS BUFFER
015B CD3C03	204		CALL	WP75	;CALL LOAD CURSOR POSITION SUBROUTINE
015E C9	205		RET		
	206				
	207				
	208				
015F 2AD387	209	ESCB:	LHLD	RCTAD	;LOAD ROWCOUNT IN H&L
0162 7D	210		MOV	A,L	;MOVE LOW BYTE OF ROWCOUNT TO A
0163 FE80	211		CPI	80H	;COMPARE BYTE WITH 80H
0165 CA6C01	212		JZ	GAMMA	;IF BYTE=80H, CONTINUE COMPARRISON
0168 CD1A03	213		CALL	ROWDN	;CALL ROWDOWN SUBROUTINE
016B C9	214		RET		;RETURN
016C 7C	215	GAMMA:	MOV	A,H	;MOVE UP BYTE OF ROWCOUNT TO A
016D FE07	216		CPI	07H	;COMPARE BYTE WITH 07H
016F CA7601	217		JZ	DELTA	;IF BYTE=07H, ROWCOUNT=LAST ROW
0172 CD1A03	218		CALL	ROWDN	;CALL ROWDOWN SUBROUTINE
0175 C9	219		RET		;RETURN
0176 CD3C03	220	DELTA:	CALL	WP75	;CALL LOAD CURSOR POSITION SUBROUTINE
0179 CD0B04	221		CALL	SCROL	;CALL SCROLL SUBROUTINE
017C C9	222		RET		;RETURN
	223				
	224				
	225				
017D 3AD287	226	ESCC:	LDA	CCTAD	;LOAD COLUMN COUNT IN A
0180 FE4F	227		CPI	4FH	;COMPARE BYTE WITH 4FH
0182 CA8901	228		JZ	ZETA	;IF BYTE=4FH, COLUMN COUNT =LAST
	229				;CHARACTER POS IN ROW
0185 CD3403	230		CALL	COLRT	;CALL COLUMN RIGHT SUBROUTINE
0188 C9	231		RET		;RETURN
0189 2AD387	232	ZETA:	LHLD	RCTAD	;LOAD ROWCOUNT IN H&L
018C 7D	233		MOV	A,L	;MOVE LOW BYTE OF ROWCOUNT TO A
018D FE80	234		CPI	80H	;COMPARE BYTE WITH 80H
018F C29B01	235		JNZ	CCTOA	;IF BYTE=80H, CONTINUE COMPARRISON
0192 7C	236		MOV	A,H	;MOVE UP BYTE OF ROWCOUNT TO A
0193 FE07	237		CPI	07H	;COMPARE BYTE WITH 07H
0195 C29B01	238		JNZ	CCTOA	;IF BYTE=07H,ROWCOUNT=LAST ROW
0198 C3A401	239		JMP	CCTOB	;JUMP TO CCTOB
019B 3E00	240	CCTOA:	MVI	A,00H	;ZERO A
019D 32D287	241		STA	CCTAD	;ZERO COLUMN COUNT
01A0 CD1A03	242		CALL	ROWDN	;CALL ROWDOWN SUBROUTINE
01A3 C9	243		RET		;RETURN
01A4 3E00	244	CCTOB:	MVI	A,00H	;ZERO A
01A6 32D287	245		STA	CCTAD	;ZERO COLUMN COUNT BUFFER
01A9 CD3C03	246		CALL	WP75	;CALL LOAD CURSOR POSITION SUBROUTINE
01AC CD0B04	247		CALL	SCROL	;CALL SCROLL SUBROUTINE
01AF C9	248		RET		;RETURN
	249				
	250				
	251				
01B0 3AD287	252	ESCD:	LDA	CCTAD	;LOAD COLUMN COUNT IN A
01B3 FE00	253		CPI	00H	;COMPARE BYTE WITH 00H
01B5 CABCO1	254		JZ	NXTA	;IF BYTE=0,COLUMN COUNT =FIRST CHAR POS IN ROW
01B8 CD2C03	255		CALL	COLLT	;CALL COLUMN LEFT SUBROUTINE
01BB C9	256		RET		;RETURN
01BC 2AD387	257	NXTA:	LHLD	RCTAD	;LOAD ROWCOUNT IN H&L
01BF 7D	258		MOV	A,L	;LOAD LOW BYTE OF ROWCOUNT IN A
01C0 FE00	259		CPI	00H	;COMPARE BYTE WITH 00H
01C2 C2CE01	260		JNZ	CCTMA	;IF BYTE=0,CONTINUE COMPARRISON
01C5 7C	261		MOV	A,H	;LOAD UP BYTE OF ROWCOUNT IN A
01C6 FE00	262		CPI	00H	;COMPARE BYTE WITH ZERO
01C8 C2CE01	263		JNZ	CCTMA	;IF BYTE=0,HOME POS CONDITION EXISTS

01CB	C3D701	264	JMP	CCTMB	; JUMP TO CCTMB
01CE	3E4F	265	CCTMA: MVI	A,4FH	; LOAD A WITH 4FH
01D0	32D287	266	STA	CCTAD	; SET COLUMN COUNT=4FH=79D
01D3	CD0803	267	CALL	ROWUP	; CALL ROWUP SUBROUTINE
01D6	C9	268	RET		; RETURN
01D7	218007	269	CCTMB: LXI	H,0780H	; LOAD H&L WITH ROWCOUNT=780H=1920D
01DA	22D387	270	SHLD	RCTAD	; SET ROWCOUNT = 1920D
01DD	3E4F	271	MVI	A,4FH	; LOAD A WITH 4FH
01DF	32D287	272	STA	CCTAD	; SET COLUMN COUNT=4FH=79D
01E2	3E18	273	MVI	A,18H	; LOAD A WITH 18H
01E4	32D587	274	STA	CURSY	; SET CURSOR Y POINTER=18H=24D
01E7	CD3C03	275	CALL	WP75	; CALL LOAD CURSOR POSITION SUBROUTINE
01EA	C9	276	RET		; RETURN
		277			
		278			
		279			
01EB	210000	280	ESCH: LXI	H,0000H	; ZERO H&L
01EE	22D387	281	SHLD	RCTAD	; SET ROWCOUNT=0
01F1	3E00	282	MVI	A,00H	; ZERO A
01F3	32D287	283	STA	CCTAD	; SET COLUMN COUNT=0
01F6	32D587	284	STA	CURSY	; SET CURSOR Y POINTER=0
01F9	CD3C03	285	CALL	WP75	; CALL LOAD CURSOR POSITION SUBROUTINE
01FC	C9	286	FET		; RETURN
		287			
		288			
		289			
01FD	2AD687	290	ESCK: LHLD	TOPAD	; LOAD TOP IN H&L
0200	EB	291	XCHG		; STORE TOP IN D&E
0201	2AD387	292	LHLD	RCTAD	; LOAD ROWCOUNT IN H&L
0204	19	293	DAD	D	; ADD TOP+ROWCOUNT, RESULT IN H&L
0205	22DE87	294	SHLD	LOCXX	; STORE RESULT IN MEM
		295			
0208	3E87	296	MVI	A,87H	; LOAD 87H IN A
020A	EC	297	CMP	H	; COMPARE H WITH 87H
020B	D21402	298	JNC	FRODO	; IF NO CARRY, CONTINUE
020E	CD2A02	299	CALL	COMRX	; IF CARRY, CALL COMPENSATION ROUTINE
0211	C32002	300	JMP	BILBO	; JUMP TO BILBO
0214	C22002	301	PRODO: JNZ	BILBO	; IF NOT EQUAL END COMPARRISON
0217	3ECF	302	MVI	A,0CFH	; LOAD CFH IN A
0219	BD	303	CMP	L	; COMPARE L WITH CFH
021A	D22002	304	JNC	BILBO	; IF NO CARRY, LOCXX LESS THAN OR EQ TO 87CFH
021D	CD2A02	305	CALL	COMRX	; IF CARRY, CALL COMPENSATION ROUTINE
0220	2ADE87	306	BILBO: LHLD	LOCXX	; LOAD LOC OF FIRST CHAR IN ROW IN H&L
0223	22E287	307	SHLD	LOCBUF	; STORE LOCXX IN BUFFER
0226	CD3204	308	CALL	FILL	; CALL FILL ROW WITH SP CHAR SUBROUTINE
0229	C9	309	RET		; RETURN
		310			
		311			
		312			
022A	2ADE87	313	COMRX: LHLD	LOCXX	; LOAD LOCXX IN H&L
022D	1130F8	314	LXI	D,0F830H	; LOAD COMPENSATION VALUE IN D&E
0230	19	315	DAD	D	; ADD D&E TO H&L
0231	22DE87	316	SHLD	LOCXX	; STORE RESULT IN LOCXX
0234	C9	317	RET		
		318			
		319			
		320			
0235	3EFO	321	ESCE: MVI	A,0FOH	; MOVE EOR CHAR TO A
0237	0619	322	MVI	B,19H	; MOVE LOOP CTR START VALUE = 19H=25D TO B
0239	115000	323	LXI	D,50H	; MOVE 80D=50H TO D&E
023C	210080	324	LXI	H,8000H	; MOVE 8000H TO H&L
		325			
023F	77	326	LOADX: MOV	M,A	; MOVE EOR CHARACTER TO MEM
0240	19	327	DAD	D	; ADD 80D=50H TO ADDRESS IN H&L
0241	05	328	DCR	B	; DECREMENT B
0242	C23F02	329	JNZ	LOADX	; CONTINUE LOOPING IF B NOT ZERO
		330			
0245	210000	331	LXI	H,0000H	; ZERO H&L
0248	22D387	332	SHLD	RCTAD	; ZERO ROWCOUNT
024B	210080	333	LXI	H,8000H	
024E	22D687	334	SHLD	TOPAD	
0251	218087	335	LXI	H,8780H	
0254	22E687	336	SHLD	BOTAD	
0257	3E00	337	MVI	A,00H	; ZERO A
0259	32D287	338	STA	CCTAD	; ZERO COLUMN COUNT
025C	32D587	339	STA	CURSY	; ZERO CURSOR Y POS
025F	32E487	340	STA	XFLG	
0262	CD3C03	341	CALL	WP75	; CALL LOAD CURSOR POSITION SUBROUTINE
0265	C9	342	RET		
		343			
		344			
		345			
0266	2AD687	346	ESCJ: LHLD	TOPAD	; LOAD TOP IN H&L
0269	EB	347	XCHG		; STORE TOP IN D&E
026A	2AD387	348	LHLD	RCTAD	; LOAD ROW COUNT IN H&L
026D	19	349	DAD	D	; ADD TOP+ROWCOUNT, YIELDING LOC OF
		350			; FIRST CHAR IN PRESENT ROW
026E	22E087	351	SHLD	LOCPR	; STORE LOCATION IN MEM
		352			

027A	C38902	357		JMP	FIN	;LINE COMPENSATION ROUTINE
027D	C28902	358	VAR:	JNZ	FIN	;JUMP TO FIN
0280	3ECF	359		MVI	A,OCFH	;IF NOT EQUAL END COMPARRISON
0282	BD	360		CMP	L	;LOAD CFH IN A
0283	D28902	361		JNC	FIN	;COMPARE L WITH CFH
0286	CDEE02	362		CALL	COMRY	;IF NO CARRY,LOCPR LESS THAN OR EQ TO 87CFH
		363				;CALL COMPENSATION ROUTINE
		364				
0289	2AD687	365	FIN:	LHLD	TOPAD	;LOAD TOP IN H&L
028C	7D	366		MOV	A,L	;MOVE L TO A
028D	FE00	367		CPI	00H	;COMPARE BYTE TO 00H
028F	C2A102	368		JNZ	TROLL	;IF NO COMPARRISON, JUMP TO TROLL
0292	7C	369		MOV	A,H	;MOVE H TO A
0293	FE80	370		CPI	80H	;COMPARE BYTE WITH 80H
0295	C2A102	371		JNZ	TROLL	;IF NO COMPARRISON, JUMP TO TROLL
0298	218087	372		LXI	H,8780H	;IF COMPARRISON,SET BOT=8780H
029B	22E687	373		SHLD	BOTAD	
029E	C3AB02	374		JMP	GNOME	;JUMP TO GNOME
02A1	11B0FF	375	TROLL:	LXI	D,OFFBOH	;LOAD -80D=OFFBOH IN D&E
02A4	2AD687	376		LHLD	TOPAD	;LOAD TOP IN H&L
02A7	19	377		DAD	D	;ADD -80D TO TOP
02A8	22E687	378		SHLD	BOTAD	
		379				
02AB	3EFO	380	GNOME:	MVI	A,OF0H	;LOAD A WITH EOR CHAR (LOOP START)
		381				
02AD	2AE087	382		LHLD	LOCPR	;LOAD LOCPR IN H&L
02B0	77	383		MOV	M,A	;MOVE EOR CHAR TO MEM
		384				
02B1	7D	385		MOV	A,L	;MOVE L TO A
02B2	FE80	386		CPI	80H	;COMPARE YTE WITH 80H
02B4	C2D502	387		JNZ	WIZAR	;IF NO COMPARRISON, JUMP TO WIZAR
02B7	7C	388		MOV	A,H	;MOVE H TO A
02B8	FE87	389		CPI	87H	;COMPARE BYTE WITH 87H
02BA	C2D502	390		JNZ	WIZAR	;IF NO COMPARRISON, JUMP TO WIZAR
		391				;IF COMPARRISON, PROCEED TO GZONK
02BD	EB	392	GZONK:	XCHG		;STORE PRESENT LOC IN D&E
02BE	2AE687	393		LHLD	BOTAD	;LOAD BOT IN H&L
02C1	7D	394		MOV	A,L	;MOVE L TO A
02C2	BB	395		CMP	E	;COMPARE E WITH A
02C3	C2CC02	396		JNZ	FUN	;IF NO COMP, JUMP TO FUN
02C6	7C	397		MOV	A,H	;MOVE H TO A
02C7	EA	398		CMP	D	;COMPARE D WITH A
02C8	C2CC02	399		JNZ	FUN	;IF NO COMP, JUMP TO FUN
		400				;IF COMPARRISON, RETURN
02CB	C9	401		RET		;RETURN
02CC	210080	402	FUN:	LXI	H,8000H	;LOAD H&L WITH 8000H
02CF	22E087	403		SHLD	LOCPR	;SET LOCPR =8000H
02D2	C3AB02	404		JMP	GNOME	
		405				
02D5	EB	406	WIZAR:	XCHG		;STORE LOCPR IN D&E
02D6	2AE687	407		LHLD	BOTAD	;LOAD BOT IN H&L
02D9	7D	408		MOV	A,L	;MOVE L TO A
02DA	BB	409		CMP	E	;COMPARE E WITH A
02DB	C2E402	410		JNZ	NUF	;IF NO COMP, JUMP TO NUF
02DE	7C	411		MOV	A,H	;MOVE H TO A
02DF	BA	412		CMP	D	;COMPARE D WITH A
02E0	C2E402	413		JNZ	NUF	;IF NO COMP, JUMP TO NUF
		414				;IF COMPARRISON, RETURN
02E3	C9	415		RET		;RETURN
02E4	215000	416	NUF:	LXI	H,50H	;LOAD 80D=50H IN H&L
02E7	19	417		DAD	D	;ADD 80D TO LOCPR (LOCPR IN D&E)
02E8	22E087	418		SHLD	LOCPR	;STORE LOCPR IN MEM
02EB	C3AB02	419		JMP	GNOME	;JUMP TO GNOME
		420				
		421				
		422				;COMPENSATION SUBROUTINE COMRY
02EE	2AE087	423	COMRY:	LHLD	LOCPR	;LOAD LOCPR IN H&L
02F1	1130F8	424		LXI	D,OF830H	;LOAD COM VALUE IN D&E
02F4	19	425		DAD	D	;ADD COMPENSATION TO LOCPR
02F5	22E087	426		SHLD	LOCPR	;STORE LOCPR IN MEM
02F8	C9	427		RET		;RETURN
		428				
		429				;LINE FEED ROUTINE
02F9	C35F01	430	CTRLJ:	JMP	ESCB	
		431				
		432				;CARRIAGE RETURN ROUTINE
		433				
02FC	3E00	434	CTRLM:	MVI	A,00H	;ZERO A
02FE	32D287	435		STA	CCTAD	;SET COLUMN COUNT=0
0301	CD3C03	436		CALL	WP75	;CALL LOAD CURSOR POSITION SUBROUTINE
0304	C9	437		RET		;RETURN
		438				
		439				;BACK SPACE ROUTINE
		440				
0305	C3B001	441	CTRLH:	JMP	ESCD	

	442			
	443			
	444			
	445			
0308	2AD387	445	ROWUP:	LHLD RCTAD ;LOAD ROWCOUNT IN H&L
030B	11B0FF	446		LXI D,OFFB0H ;MOVE -80D=OFFB0H (2'S COMP) TO D&E
030E	19	447		DAD D ;ADD -80D TO ROWCOUNT
030F	22D387	448		SHLD RCTAD ;STORE RESULT IN ROWCOUNT BUFFER
		449		
0312	21D587	450		LXI H,CURSY ;LOAD CURSOR Y POINTER ADDRESS IN H&L
0315	35	451		DCR M ;DECREMENT CURSOR Y POINTER
0316	CD3C03	452		CALL WP75 ;CALL LOAD CURSOR POSITION SUBROUTINE
0319	C9	453		RET ;RETURN
		454		
		455		
		456		
		457		
031A	2AD387	457	ROWDN:	LHLD RCTAD ;LOAD ROWCOUNT IN H&L
031D	115000	458		LXI D,50H ;MOVE +80D=50H TO D&E
0320	19	459		DAD D ;ADD +80D TO ROWCOUNT
0321	22D387	460		SHLD RCTAD ;STORE RESULT IN ROWCOUNT
0324	21D587	461		LXI H,CURSY ;LOAD CURSOR Y POINTER ADDRESS IN H&L
0327	34	462		INR M ;INCREMENT CURSOR Y POINTER
0328	CD3C03	463		CALL WP75 ;CALL LOAD CURSOR POSITION SUBROUTINE
032B	C9	464		RET ;RETURN
		465		
		466		
		467		
032C	21D287	468	COLLT:	LXI H,CCTAD ;LOAD COLUMN COUNT ADDRESS IN H&L
032F	35	469		DCR M ;DECREMENT COLUMN COUNT
0330	CD3C03	470		CALL WP75 ;CALL LOAD CURSOR POSITION SUBROUTINE
0333	C9	471		RET ;RETURN
		472		
		473		
		474		
		475		
0334	21D287	475	COLRT:	LXI H,CCTAD ;LOAD COLUMN COUNT ADDRESS IN H&L
0337	34	476		INR M ;INCREMENT COLUMN COUNT
0338	CD3C03	477		CALL WP75 ;CALL LOAD CURSOR POSITION SUBROUTINE
033B	C9	478		RET ;RETURN
		479		
		480		
		481		
		482		
033C	3E80	482	WP75:	MVI A,80H ;LOAD A WITH 80H, LOAD CURSOR POSITION COMMAND
033E	D35F	483		OUT CROM
0340	3AD287	484		LDA CCTAD ;LOAD A WITH CURSOR X POSITION
0343	D35E	485		OUT CRDAT
0345	3AD587	486		LDA CURSY ;LOAD A WITH CURSOR Y POSITION
0348	D35E	487		OUT CRDAT
034A	C9	488		RET ;RETURN
		489		
		490		
		491		
		492		
		493		
034B	3AD287	493	DISPL:	LDA CCTAD ;LOAD COLUMN COUNT IN H&L
034E	FE4F	494		CPI 4FH ;COMPARE BYTE WITH 4FH=79D
0350	CA5A03	495		JZ CTA ;IF BYTE=4FH,COLUMN COUNT=LAST CHAR-
		496		
0353	CD7E03	497		CALL DIS1 ;ACTER IN ROW
0356	CDBB03	498		CALL DISA ;CALL DIS1 SUBROUTINE
0359	C9	499		RET ;CALL DISA SUBROUTINE
035A	2AD387	500	CTA:	LHLD RCTAD ;LOAD ROWCOUNT IN H&L
035D	7D	501		MOV A,L ;LOAD LOW BYTE OF ROWCOUNT IN H&L
035E	FE80	502		CPI 80H ;COMPARE BYTE WITH 80H
0360	CA6A03	503		JZ CTB ;IF BYTE=80H,CONTINUE COMPARRISON
0363	CD7E03	504		CALL DIS1 ;CALL DIS1 SUBROUTINE
0366	CDC303	505		CALL DISB ;CALL DISB SUBROUTINE
0369	C9	506		RET ;RETURN
036A	7C	507	CTB:	MOV A,H ;MOVE UP BYTE OF ROWCOUNT TO H&L
036B	FE07	508		CPI 07H ;COMPARE BYTE WITH 07H
036D	CA7703	509		JZ CTC ;IF BYTE=07H,END OF DISPLAY COND EXISTS
0370	CD7E03	510		CALL DIS1 ;CALL DIS1 SUBROUTINE
0373	CDC303	511		CALL DISB ;CALL DISB SUBROUTINE
0376	C9	512		RET ;RETURN
0377	CD7E03	513	CTC:	CALL DIS1 ;CALL DIS1 SUBROUTINE
037A	CDDA03	514		CALL DISC ;CALL DISC SUBROUTINE
037D	C9	515		RET ;RETURN
		516		
		517		
		518		
		519		
037E	2AD687	519	DIS1:	LHLD TOPAD ;LOAD TOP IN H&L
0381	EB	520		XCHG ;STORE TOP IN D&E
0382	2AD387	521		LHLD RCTAD ;LOAD ROWCOUNT IN H&L
0385	19	522		DAD D ;ADD TOP+ROWCOUNT, RESULT IN H&L
0386	22DA87	523		SHLD LOC01 ;STORE LOCATION OF FIRST CHAR IN ROW
0389	EB	524		XCHG ;STORE TOP+ROWCOUNT IN D&E
038A	210000	525		LXI H,0000H ;ZERO H&L
038D	3AD287	526		LDA CCTAD ;LOAD COLUMN COUNT IN A
0390	6F	527		MOV L,A ;MOVE COLUMN COUNT TO L
0391	19	528		DAD D ;CALCULATE LOCATION=
		529		
0392	22D887	530		SHLD LOCAD ;TOP+ROWCOUNT+COLUMN COUNT,RESULT IN H&L
0395	3E87	531		MVI A,87H ;STORE LOCATION IN MEMORY
				LOAD 87H IN A

0397 BC	532	CMP	H	;COMPARE H WITH 87H
0398 D2A103	533	JNC	NXTCM	;IF NO CARRY, CONTINUE COMPARRISON
039B CDE603	534	CALL	COMRT	;IF CARRY, CALL COMPENSATION ROUTINE
039E C3AD03	535	JMP	XSTAD	;JUMP TO XSTAD
03A1 C2AD03	536	JNZ	XSTAD	;IF NOT EQUAL, END COMPARRISON
03A4 3ECF	537	MVI	A, OCFH	;LOAD OCFH IN A
03A6 BD	538	CMP	L	;COMPARE L WITH OCFH
03A7 D2AD03	539	JNC	XSTAD	;IF NO CARRY, LOCATION LESS THAN
	540			;OR EQUAL TO 87CFH
03AA CDE603	541	CALL	COMRT	;IF CARRY, CALL COMPENSATION ROUTINE
03AD CDFB03	542	CALL	EORT	;CALL END OF ROW CHAR TEST ROUTINE
03B0 21E587	543	LXI	H, USCHR	;LOAD USART CHAR ADD IN H&L
03B3 7E	544	MOV	A, M	;MOVE USART CHAR TO A
03B4 E63F	545	ANI	3FH	;MASK OFF UPPER 2 BITS OF CHAR
03B6 2AD887	546	LHLD	LOCAD	;LOAD LOCATION IN H&L
03B9 77	547	MOV	M, A	;MOVE CHARACTER TO CHARACTER
	548			;LOCATION IN DISPLAY MEMORY
03BA C9	549	RET		;RETURN
	550			
	551			;SUBROUTINE DISA
	552			
03BB 21D287	553	LXI	H, CCTAD	;LOAD COLUMN COUNT ADD IN H&L
03BE 34	554	INR	M	;INCREMENT COLUMN COUNT
03BF CD3C03	555	CALL	WP75	;CALL LOAD CURSOR POSITION SUBROUTINE
03C2 C9	556	RET		;RETURN
	557			
	558			;SUBROUTINE DISP
	559			
03C3 3E00	560	MVI	A, 00H	;ZERO A
03C5 32D287	561	STA	CCTAD	;ZERO COLUMN COUNT
03C8 2AD387	562	LHLD	RCTAD	;LOAD ROWCOUNT IN H&L
03CB 115000	563	LXI	D, 50H	;LOAD 80D=50H IN D&E
03CE 19	564	DAD	D	;ADD +80 TO ROWCOUNT
03CF 22D387	565	SHLD	RCTAD	;STORE ROWCOUNT IN MEMORY
03D2 21D587	566	LXI	H, CURSY	;LOAD CURSOR Y POSITION ADDRESS IN H&L
03D5 34	567	INR	M	;INCREMENT CURSOR Y POSITION
03D6 CD3C03	568	CALL	WP75	;CALL LOAD CURSOR POSITION SUBROUTINE
03D9 C9	569	RET		;RETURN
	570			
	571			;SUBROUTINE DISC
	572			
03DA 3E00	573	MVI	A, 00H	;ZERO A
03DC 32D287	574	STA	CCTAD	;ZERO COLUMN COUNT
03DF CD3C03	575	CALL	WP75	;CALL LOAD CURSOR POSITION SUBROUTINE
03E2 CDB0B4	576	CALL	SCROL	
03E5 C9	577	RET		;RETURN
	578			
	579			;ADDRESS COMPENSATION SUBROUTINE
	580			
03E6 2AD887	581	LHLD	LOCAD	;LOAD CHARACTER LOCATION
03E9 1130F8	582	LXI	D, 0F830H	;LOAD COMPENSATION VALUE IN D&E
03EC 19	583	DAD	D	;ADD COMPENSATION TO LOCATION
03ED 22D887	584	SHLD	LOCAD	;STORE MODIFIED LOCATION IN MEMORY
	585			
03F0 2ADA87	586	LHLD	LOC01	;LOAD LOCATION OF FIRST CHAR
	587			;IN ROW IN H&L
03F3 1130F8	588	LXI	D, 0F830H	;LOAD COMPENSATION VALUE IN H&L
03F6 19	589	DAD	D	;ADD COMPENSATION TO LOC01
03F7 22DA87	590	SHLD	LOC01	;STORE MODIFIED LOC01 IN MEMORY
03FA C9	591	RET		;RETURN
	592			
	593			;END OF ROW TEST ROUTINE
	594			
03FB 2ADA87	595	LHLD	LOC01	;LOAD LOCATION OF FIRST CHAR
	596			;IN ROW IN H&L
03FE 7E	597	MOV	A, M	;MOVE FIRST CHAR IN ROW TO A REG
03FF FEFO	598	CPI	0F0H	;COMPARE CHAR WITH 0F0 (END OF ROW CHAR)
0401 C20A04	599	JNZ	XIT	;IF NO COMPARRISON, EXIT
0404 22E287	600	SHLD	LOCBUF	;STORE FIRST CHAR IN ROW ADD IN LOCBUF
0407 CD3204	601	CALL	FILL	;CALL FILL ROW WITH SPACE CODES SUBROUTINE
040A C9	602	RET		;RETURN
	603			
	604			;SCROLL SUBROUTINE
	605			
040B 2AD687	606	LHLD	TOPAD	;LOAD TOP IN H&L
040E 22E287	607	SHLD	LOCBUF	;STORE FIRST CHAR IN ROW ADD IN LOCBUF
0411 CD3204	608	CALL	FILL	;CALL FILL ROW WITH SPACE CODES SUBROUTINE
0414 2AD687	609	LHLD	TOPAD	;MOVE TOP TO H&L
0417 7D	610	MOV	A, L	;MOVE LOWER BYTE OF TOP TO A
0418 FE80	611	CPI	80H	;COMPARE TOP WITH MAX VALUE
041A C22A04	612	JNZ	DUCK	;IF NO COMPARRISON EXISTS, CONTINUE SCROL
041D 7C	613	MOV	A, H	;MOVE UPPER BYTE OF TOP TO A
041E FE87	614	CPI	87H	;COMPARE TOP WITH MAX VALUE
0420 C22A04	615	JNZ	DUCK	;IF NO COMPARRISON EXISTS, CONTINUE SCROL
	616			;IF COMPARRISON, TOP=MAX VALUE=8780H
0423 210080	617	LXI	H, 8000H	;IF COMPARRISON, MODIFY TOP TO TOP=8000H
0426 22D687	618	SHLD	TOPAD	;STORE MODIFIED TOPAD IN MEMORY
0429 C9	619	RET		;RETURN
042A 115000	620	LXI	D, 50H	;MOVE 80D=50H TO D&E

042D 19	621	DAD	D	;ADD 80D=50H TO TOP
042E 22D687	622	SHLD	TOPAD	;STORE MODIFIED TOPAD IN MEMORY
0431 C9	623	RET		;RETURN
	624			
	625			
	626			
0432 2AE287	627	FILL: LHL	LOCBUF	;LOAD LOCATION OF FIRST CHAR IN ROW
	628			;OR FIRST CHAR IN TOP ROW IN H&L
0435 115000	629	LXI	D,50H	;LOAD 80D=50H IN D&E
0438 19	630	DAD	D	;CALCULATE LOCATION OF LAST CHAR IN ROW
0439 22DC87	631	SHLD	LOC80	;STORE LOCATION OF LAST CHAR IN ROW IN MEMORY
043C 012020	632	LXI	B,2020H	;LOAD SPACE CHARACTERS IN B&C
043F 210000	633	LXI	H,0000H	;ZERO H&L
0442 39	634	DAD	SP	;ADD SP TO H&L, TRANSFERRING SP TO H&L
0443 EB	635	XCHG		;STORE STACK POINTER IN D&E
0444 2ADC87	636	LHL	LOC80	;LOAD LOCATION OF LAST CHAR IN ROW IN H&L
0447 F9	637	SPHL		;LOAD LAST CHAR LOCATION IN SP
	638			
0448 C5	639	PUSH	B	;EXECUTE THE LIST OF PUSH B COMMANDS TO
0449 C5	640	PUSH	B	;FILL THE LINE WITH BLANK CHARACTERS
044A C5	641	PUSH	B	
044B C5	642	PUSH	B	
044C C5	643	PUSH	B	
044D C5	644	PUSH	B	
044E C5	645	PUSH	B	
044F C5	646	PUSH	B	
0450 C5	647	PUSH	B	
0451 C5	648	PUSH	B	
0452 C5	649	PUSH	B	
0453 C5	650	PUSH	B	
0454 C5	651	PUSH	B	
0455 C5	652	PUSH	B	
0456 C5	653	PUSH	B	
0457 C5	654	PUSH	B	
0458 C5	655	PUSH	B	
0459 C5	656	PUSH	B	
045A C5	657	PUSH	B	
045B C5	658	PUSH	B	
045C C5	659	PUSH	B	
045D C5	660	PUSH	B	
045E C5	661	PUSH	B	
045F C5	662	PUSH	B	
0460 C5	663	PUSH	B	
0461 C5	664	PUSH	B	
0462 C5	665	PUSH	B	
0463 C5	666	PUSH	B	
0464 C5	667	PUSH	B	
0465 C5	668	PUSH	B	
0466 C5	669	PUSH	B	
0467 C5	670	PUSH	B	
0468 C5	671	PUSH	B	
0469 C5	672	PUSH	B	
046A C5	673	PUSH	B	
046B C5	674	PUSH	B	
046C C5	675	PUSH	B	
046D C5	676	PUSH	B	
046E C5	677	PUSH	B	
046F C5	678	PUSH	B	
0470 EB	679	XCHG		;STACK POINTER TRANSFERRED TO H&L
0471 F9	680	SPHL		;RESTORE STACK
0472 C9	681	RET		;RETURN
	682			
	683			
	684			
	685			
	686			
	687			
	688			
	689			
0473 3E00	690	RT75: MVI	A,MDC57	;MOVE MODE CLEAR COMMAND TO A
0475 D348	691	OUT	PMD57	;OUTPUT MODE CLEAR COMMAND TO 8257
	692			
0477 2AD687	693	LHL	TOPAD	;LOAD TOP IN H&L
047A 7D	694	MOV	A,L	;LOAD CH 2 START ADD, LOW BYTE, IN A
047B D344	695	OUT	PC2SA	;OUTPUT CH 2 START ADD TO 8257
047D 7C	696	MOV	A,H	;LOAD CH 2 START ADD, UP BYTE, IN A
047E D344	697	OUT	PC2SA	;OUTPUT CH 2 START ADD TO 8257
	698			
0480 7D	699	MOV	A,L	;LOAD LOW BYTE OF TOP IN A
0481 2F	700	CMA		;COMPLEMENT A
0482 6F	701	MOV	L,A	;LOAD COMPLEMENTED VALUE IN L
0483 7C	702	MOV	A,H	;LOAD UP BYTE OF TOP IN A
0484 2F	703	CMA		;COMPLEMENT A
0485 67	704	MOV	H,A	;LOAD COMPLEMENTED VALUE IN H
0486 23	705	INX	H	;INCREMENT H&L, YIELDING 2'S COMPLEMENT
	706			;OF TOP IN A
0487 11CF87	707	LXI	D,87CFH	;LOAD 87CFH IN D&E
048A 19	708	DAD	D	;ADD H&L TO D&E, YIELDING 87CFH-TOP
048B 110080	709	LXI	D,8000H	;LOAD D&E WITH 8000H
048E 19	710	DAD	D	;ADD 8000H TO 87CF-TOP

048F 7D	711	MOV	A,L	;MOVE LOW BYTE OF CH 2 TC TO A
0490 D345	712	OUT	PC2TC	;OUTPUT CH 2 TC TO 8257
0492 7C	713	MOV	A,H	;MOVE UP BYTE OF CH 2 TC TO A
0493 D345	714	OUT	PC2TC	;OUTPUT CH 2 TC TO 8257
	715			
0495 210080	716	LXI	H,8000H	;LOAD 8000H IN H&L
0498 7D	717	MOV	A,L	;MOVE LOW BYTE OF CH 3 START ADD TO A
0499 D346	718	OUT	PC3SA	;OUTPUT CH 3 START ADD TO 8257
049B 7C	719	MOV	A,H	;MOVE UP BYTE OF CH 3 START ADD TO A
049C D346	720	OUT	PC3SA	;OUTPUT CH 3 START ADD TO 8257
	721			
049E 21CF87	722	LXI	H,87CFH	;LOAD CH 3 TC VALUE IN H&L
04A1 7D	723	MOV	A,L	;MOVE L TO A
04A2 D347	724	OUT	PC3TC	;OUTPUT CH 3 TC TO 8257
04A4 7C	725	MOV	A,H	;MOVE H TO A
04A5 D347	726	OUT	PC3TC	;OUTPUT CH 3 TC TO 8257
04A7 3E84	727	MVI	A,MDS57	;LOAD A WITH MODE SET VALUE
04A9 D348	728	OUT	PMD57	;OUTPUT MODE SET TO 8257
	729			
	730			
	731			
	732			
	733			
04AB DB6F	733	KPOLL:	IN KCOM	;INPUT FIFO STATUS
04AD E607	734		ANI 07H	;MASK STATUS, SAVE BITS 0-2
04AF CAB504	735		JZ ZIP	;TEST FOR CHARACTER PRESENT
04B2 CDB604	736		CALL XMIT	;CALL CHARACTER TRANSMIT ROUTINE
04B5 C9	737		RET	;RETURN
	738			
	739			
	740			
	741			
04B6 DB6E	741	XMIT:	IN KDAT	;INPUT FIFO CHARACTER
04B8 EEC6	742		XRI 0C0H	;INVERT TOP 2 BITS
04BA 21F804	743		LXI H,BSET3	;LOAD BASE ADD OF TABLE 3 IN H&L
04BD 110000	744		LXI D,0000H	;ZERO D&E
04C0 5F	745		MOV E,A	;LOAD E WITH CHARACTER FROM FIFO
04C1 19	746		DAD D	;CALCULATE ADD IN LOOKUP TABLE
	747			;CONTAINING ASCII CHARACTERS
	748			;CORRESPONDING TO KEY POSITION IN MATRIX
04C2 DBFB	749	USZ:	IN CNCTL	;INPUT USART STATUS
04C4 E601	750		ANI 01H	;MASK STATUS, SAVE TRANSMITTER READY BIT
04C6 CAC204	751		JZ USZ	;TEST READY BIT
04C9 7E	752		MOV A,M	;MOVE ASCII CHAR TO A
04CA E67F	753		ANI 7FH	;MASK BIT 7
04CC D3FA	754		OUT CNOUT	;OUTPUT CHAR FROM USART
04CE C9	755		RET	;RETURN
	756			
	757			
	758			
	759			
04CF C9	759	DUMY:	RET	;RETURN
	760			
	761			
	762			
	763			
	764			
	765			
	766			
	767			
	768			
04D0 CF04	768	BSET1:	DW DUMY	
04D2 3901	769		DW ESCA	
04D4 5F01	770		DW ESCB	
04D6 7D01	771		DW ESCC	
04D8 B001	772		DW ESCD	
04DA 3502	773		DW ESCE	
04DC CF04	774		DW DUMY	
04DE CF04	775		DW DUMY	
04E0 EB01	776		DW ESCH	
04E2 CF04	777		DW DUMY	
04E4 6602	778		DW ESCJ	
04E6 FD01	779		DW ESCK	
04E8 CF04	780		DW DUMY	
04EA CF04	781		DW DUMY	
04EC CF04	782		DW DUMY	
04EE CF04	783		DW DUMY	
	784			
	785			
04F0 0503	786	BSET2:	DW CTRLH	
04F2 F902	787		DW CTRLJ	
04F4 FC02	788		DW CTRLM	
04F6 CF04	789		DW DUMY	
	790			
	791			
04F8 30	792	BSET3:	DB 30H	;DUMMY CHARACTER
04F9 30	793		DB 30H	
04FA 30	794		DB 30H	
04FB 30	795		DB 30H	
04FC 30	796		DB 30H	
04FD 30	797		DB 30H	
04FE 30	798		DB 30H	
04FF 30	799		DB 30H	

0500	30	800	DE	30H
0501	30	801	DE	30H
0502	30	802	DE	30H
0503	30	803	DE	30H
0504	30	804	DE	30H
0505	30	805	DE	30H
0506	30	806	DE	30H
0507	30	807	DE	30H
0508	1B	808	DE	1BH
0509	0A	809	DE	0AH
050A	2C	810	DE	2CH
050B	0D	811	DE	0DH
050C	20	812	DE	20H
050D	7F	813	DE	7FH
050E	2E	814	DE	2EH
050F	2F	815	DE	2FH
0510	5A	816	DE	5AH
0511	58	817	DE	58H
0512	4D	818	DE	4DH
0513	56	819	DE	56H
0514	30	820	DE	30H
0515	43	821	DE	43H
0516	4E	822	DE	4EH
0517	42	823	DE	42H
0518	30	824	DE	30H
0519	2D	825	DE	2DH
051A	4F	826	DE	4FH
051B	4C	827	DE	4CH
051C	39	828	DE	39H
051D	3A	829	DE	3AH
051E	50	830	DE	50H
051F	3B	831	DE	3BH
0520	53	832	DE	53H
0521	44	833	DE	44H
0522	4B	834	DE	4BH
0523	47	835	DE	47H
0524	41	836	DE	41H
0525	46	837	DE	46H
0526	4A	838	DE	4AH
0527	48	839	DE	48H
0528	57	840	DE	57H
0529	45	841	DE	45H
052A	49	842	DE	49H
052B	54	843	DE	54H
052C	51	844	DE	51H
052D	52	845	DE	52H
052E	55	846	DE	55H
052F	59	847	DE	59H
0530	32	848	DE	32H
0531	35	849	DE	35H
0532	38	850	DE	38H
0533	35	851	DE	35H
0534	31	852	DE	31H
0535	34	853	DE	34H
0536	37	854	DE	37H
0537	36	855	DE	36H
0538	30	856	DE	30H
0539	30	857	DE	30H
053A	30	858	DE	30H
053B	30	859	DE	30H
053C	30	860	DE	30H
053D	30	861	DE	30H
053E	30	862	DE	30H
053F	30	863	DE	30H
0540	30	864	DE	30H
0541	30	865	DE	30H
0542	30	866	DE	30H
0543	30	867	DE	30H
0544	30	868	DE	30H
0545	30	869	DE	30H
0546	30	870	DE	30H
0547	30	871	DE	30H
0548	30	872	DE	30H
0549	30	873	DE	30H
054A	3C	874	DE	3CH
054B	30	875	DE	30H
054C	30	876	DE	30H
054D	30	877	DE	30H
0517	42	823	DE	42H
0518	30	824	DE	30H
0519	2D	825	DE	2DH
051A	4F	826	DE	4FH
051B	4C	827	DE	4CH
051C	39	828	DE	39H
051D	3A	829	DE	3AH
051E	50	830	DE	50H
051F	3B	831	DE	3BH
0520	53	832	DE	53H
0521	44	833	DE	44H
0522	4B	834	DE	4BH

```

; ESC
; LF
; CR
; SP
; DEL
; /
; Z
; X
; M
; V
; C
; N
; E
; O
; -
; O
; L
; 9
; :
; P
; :
; S
; D
; K
; G
; A
; F
; J
; H
; W
; E
; I
; T
; O
; R
; U
; Y
; 2
; 3
; 4
; 5
; 6
; 7
; 8

```

```

; <

```

```

; B
; O
; -
; O
; L
; 9
; :
; P
; :
; S
; D
; K

```

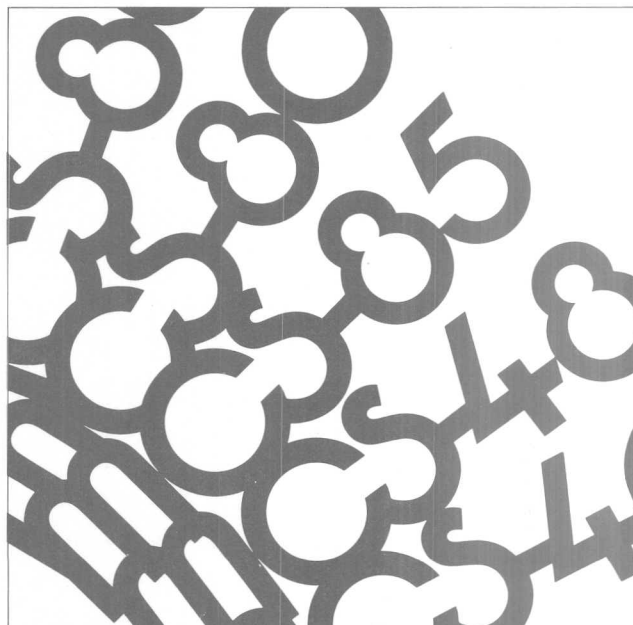
0523	47	835	DB	47H	GA
0524	41	836	DB	41H	F
0525	46	837	DB	46H	J
0526	4A	838	DB	4AH	H
0527	48	839	DB	48H	W
0528	57	840	DB	57H	E
0529	45	841	DB	45H	I
052A	49	842	DB	49H	T
052B	54	843	DB	54H	O
052C	51	844	DB	51H	R
052D	52	845	DB	52H	U
052E	55	846	DB	55H	Y
052F	59	847	DB	59H	Z
0530	32	848	DB	32H	3
0531	33	849	DB	33H	8
0532	38	850	DB	38H	5
0533	35	851	DB	35H	1
0534	31	852	DB	31H	4
0535	34	853	DB	34H	7
0536	37	854	DB	37H	6
0537	36	855	DB	36H	
0538	30	856	DB	30H	
0539	30	857	DB	30H	
053A	30	858	DB	30H	
053B	30	859	DB	30H	
053C	30	860	DB	30H	
053D	30	861	DB	30H	
053E	30	862	DB	30H	
053F	30	863	DB	30H	
0540	30	864	DB	30H	
0541	30	865	DB	30H	
0542	30	866	DB	30H	
0543	30	867	DB	30H	
0544	30	868	DB	30H	
0545	30	869	DB	30H	
0546	30	870	DB	30H	
0547	30	871	DB	30H	
0548	30	872	DB	30H	
0549	30	873	DB	30H	
054A	3C	874	DB	3CH	<
054B	30	875	DB	30H	
054C	30	876	DB	30H	
054D	30	877	DB	30H	
054E	3E	878	DB	3EH	>
054F	3F	879	DB	3FH	?
0550	30	880	DB	30H	
0551	30	881	DB	30H	
0552	5D	882	DB	5DH	]
0553	30	883	DB	30H	
0554	30	884	DB	30H	
0555	30	885	DB	30H	
0556	30	886	DB	30H	
0557	30	887	DB	30H	
0558	30	888	DB	30H	
0559	3D	889	DB	3DH	=
055A	30	890	DB	30H	
055B	5C	891	DB	5CH	\
055C	29	892	DB	29H	)
055D	2A	893	DB	2AH	*
055E	30	894	DB	30H	
055F	2B	895	DB	2BH	;
0560	30	896	DB	30H	+
0561	30	897	DB	30H	
0562	5B	898	DB	5BH	[
0563	30	899	DB	30H	
0564	30	900	DB	30H	
0565	30	901	DB	30H	
0566	30	902	DB	30H	
0567	30	903	DB	30H	
0568	30	904	DB	30H	
0569	30	905	DB	30H	
056A	30	906	DB	30H	
056B	30	907	DB	30H	
056C	30	908	DB	30H	
056D	30	909	DB	30H	
056E	30	910	DB	30H	
056F	30	911	DB	30H	
0570	22	912	DB	22H	"
0571	23	913	DB	23H	#
0572	28	914	DB	28H	(
0573	25	915	DB	25H	)
0574	21	916	DB	21H	!
0575	24	917	DB	24H	\$
0576	27	918	DB	27H	+
0577	26	919	DB	26H	&
0578	30	920	DB	30H	
0579	30	921	DB	30H	
057A	30	922	DB	30H	
057B	30	923	DB	30H	
057C	30	924	DB	30H	

057D 30	925	DB	30H	
057E 30	926	DB	30H	
057F 30	927	DB	30H	
0580 30	928	DB	30H	
0581 30	929	DB	30H	
0582 30	930	DB	30H	
0583 30	931	DB	30H	
0584 30	932	DB	30H	
0585 30	933	DB	30H	
0586 30	934	DB	30H	
0587 30	935	DB	30H	
0588 30	936	DB	30H	
0589 30	937	DB	30H	
058A 30	938	DB	30H	
058B 30	939	DB	30H	
058C 30	940	DB	30H	
058D 30	941	DB	30H	
058E 30	942	DB	30H	
058F 30	943	DB	30H	
0590 1A	944	DB	1AH	: SUB
0591 1B	945	DB	1BH	: CAN
0592 0D	946	DB	0DH	: CR
0593 16	947	DB	16H	: SYN
0594 30	948	DB	30H	
0595 03	949	DB	03H	: ETX
0596 0E	950	DB	0EH	: SO
0597 02	951	DB	02H	: STX
0598 30	952	DB	30H	
0599 1F	953	DB	1FH	: US
059A 0F	954	DB	0FH	: SI
059B 0C	955	DB	0CH	: FF
059C 30	956	DB	30H	
059D 30	957	DB	30H	
059E 10	958	DB	10H	: DLE
059F 30	959	DB	30H	
05A0 13	960	DB	13H	: DC3
05A1 04	961	DB	04H	: EOT
05A2 0B	962	DB	0BH	: VT
05A3 07	963	DB	07H	: BEL
05A4 01	964	DB	01H	: SOH
05A5 06	965	DB	06H	: ACK
05A6 0A	966	DB	0AH	: LF
05A7 08	967	DB	08H	: BS
05A8 17	968	DB	17H	: ETB
05A9 05	969	DB	05H	: ENQ
05AA 09	970	DB	09H	: HT
05AB 14	971	DB	14H	: DC4
05AC 11	972	DB	11H	: DC1
05AD 12	973	DB	12H	: DC2
05AE 15	974	DB	15H	: NAK
05AF 19	975	DB	19H	: EM
05B0 30	976	DB	30H	
05B1 30	977	DB	30H	
05B2 30	978	DB	30H	
05B3 30	979	DB	30H	
05B4 30	980	DB	30H	
05B5 30	981	DB	30H	
05B6 30	982	DB	30H	
05B7 30	983	DB	30H	
05B8 30	984	DB	30H	
05B9 30	985	DB	30H	
05BA 30	986	DB	30H	
05BB 30	987	DB	30H	
05BC 30	988	DB	30H	
05BD 30	989	DB	30H	
05BE 30	990	DB	30H	
05BF 30	991	DB	30H	
05C0 30	992	DB	30H	
05C1 30	993	DB	30H	
05C2 30	994	DB	30H	
05C3 30	995	DB	30H	
05C4 30	996	DB	30H	
05C5 30	997	DB	30H	
05C6 30	998	DB	30H	
05C7 30	999	DB	30H	
05C8 30	1000	DB	30H	
05C9 30	1001	DB	30H	
05CA 30	1002	DB	30H	
05CB 30	1003	DB	30H	
05CC 30	1004	DB	30H	
05CD 30	1005	DB	30H	
05CE 30	1006	DB	30H	
05CF 30	1007	DB	30H	
05D0 30	1008	DB	30H	
05D1 30	1009	DB	30H	
05D2 1D	1010	DB	1DH	: GS
05D3 30	1011	DB	30H	
05D4 30	1012	DB	30H	
05D5 30	1013	DB	30H	

05DB 1C	1019	DB	1CH	; FS
05DC 30	1020	DB	30H	
05DD 30	1021	DB	30H	
05DE 30	1022	DB	30H	
05DF 30	1023	DB	30H	
05E0 30	1024	DB	30H	
05E1 30	1025	DB	30H	
05E2 1B	1026	DB	1BH	; ESC
05E3 30	1027	DB	30H	
05E4 30	1028	DB	30H	
05E5 30	1029	DB	30H	
05E6 30	1030	DB	30H	
05E7 30	1031	DB	30H	
05E8 30	1032	DB	30H	
05E9 30	1033	DB	30H	
05EA 30	1034	DB	30H	
05EB 30	1035	DB	30H	
05EC 30	1036	DB	30H	
05ED 30	1037	DB	30H	
05EE 30	1038	DB	30H	
05EF 30	1039	DB	30H	
05F0 30	1040	DB	30H	
05F1 30	1041	DB	30H	
05F2 30	1042	DB	30H	
05F3 30	1043	DB	30H	
05F4 30	1044	DB	30H	
05F5 30	1045	DB	30H	
05F6 30	1046	DB	30H	
05F7 30	1047	DB	30H	
	1048	...		
	1049	...		
	1050	...		
	1051	...		
	1052	...		
	1053	...		
	1054	...		
87D2	1055	ORG	87D2H	
0001	1056	CCTAD: DS	1	
0002	1057	RCTAD: DS	2	
0001	1058	CURSY: DS	1	
0002	1059	TOPAD: DS	2	
0002	1060	LOCAD: DS	2	
0002	1061	LOC01: DS	2	
0002	1062	LOC80: DS	2	
0002	1063	LOCXX: DS	2	
0002	1064	LOCPR: DS	2	
0002	1065	LOCBUF: DS	2	
0001	1066	XFLG: DS	1	
0001	1067	USCHR: DS	1	
0002	1068	BOTAD: DS	2	
	1069	END		

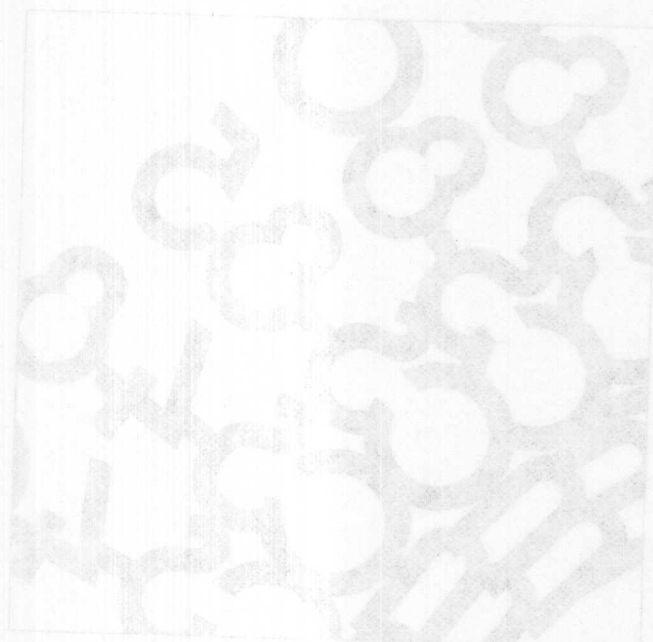
# APPENDIX 1

## ARTICLE REPRINTS



APPENDIX 1

ARTICLE 3 REPRINTS



# Slave microcomputer lightens main microprocessor load

by Don Phillips and Allen Goodman, Intel Corp., Santa Clara, Calif.

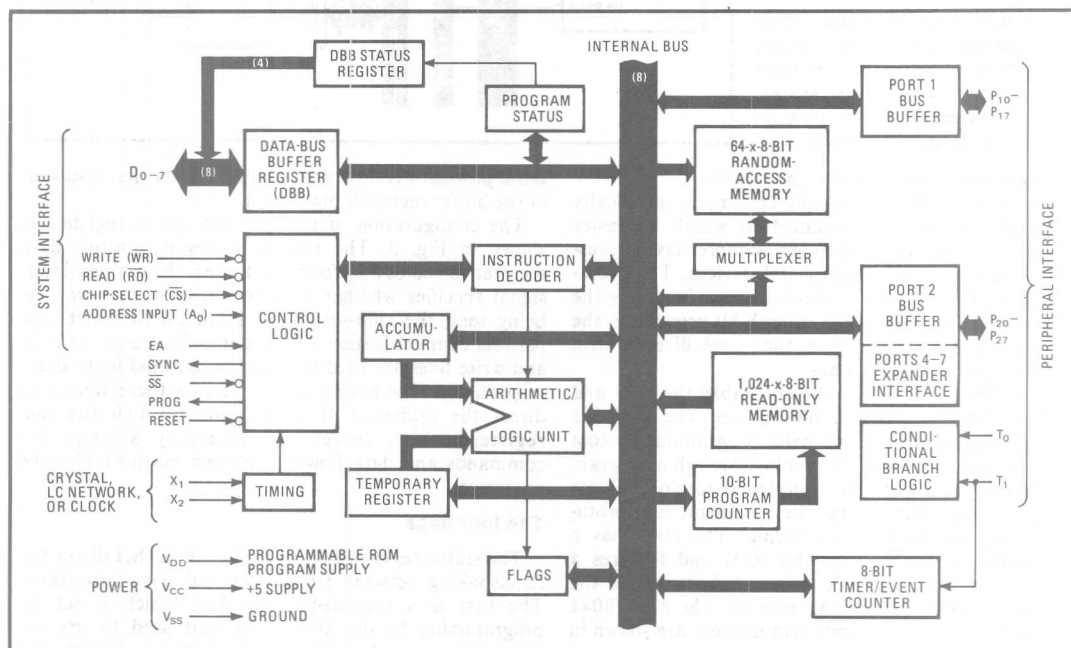
□ Peripheral devices for microprocessors are growing in number and complexity to the point where they are taxing the processor's time and memory. Nor do simple interface adapters that contain no intelligence of their own lighten the burden of managing such peripheral equipment as floppy disks, cathode-ray-tube displays, and keyboards. What can save the day for the central processing unit is a new class of peripheral controllers: intelligent microcomputer-based universal peripheral interface chips.

In essence, what the UPI microcomputer does is act as a slave processor to the main-system CPU. With a built-in processor and memory, it greatly eases the handling of real-time tasks such as controlling printers, encoding keyboards, and multiplexing displays. In fact, entire control algorithms can be programmed locally in the slave processor, instead of taxing the limited memory

space and execution time of the main system. Moreover, the device substantially increases the overall efficiency of a system, since two processors—the central CPU and the slave UPI device—are working in parallel.

## A peripheral controller

In operation, the UPI microcomputer acts as a peripheral controller rather than just an interface adapter. Its architecture, detailed in Fig. 1, is similar to the recently introduced 8048 one-chip microcomputer: it has an 8-bit CPU, 64 bytes of random-access memory, 1,024 bytes of read-only memory, a timer/counter, and 18 input/output lines. In fact, the device executes the same basic set of instructions as does the 8048, except for special tailoring of data-bus operations to better suit control applications. The difference is that the new peripheral-controlling microcomputer is designed to function as a

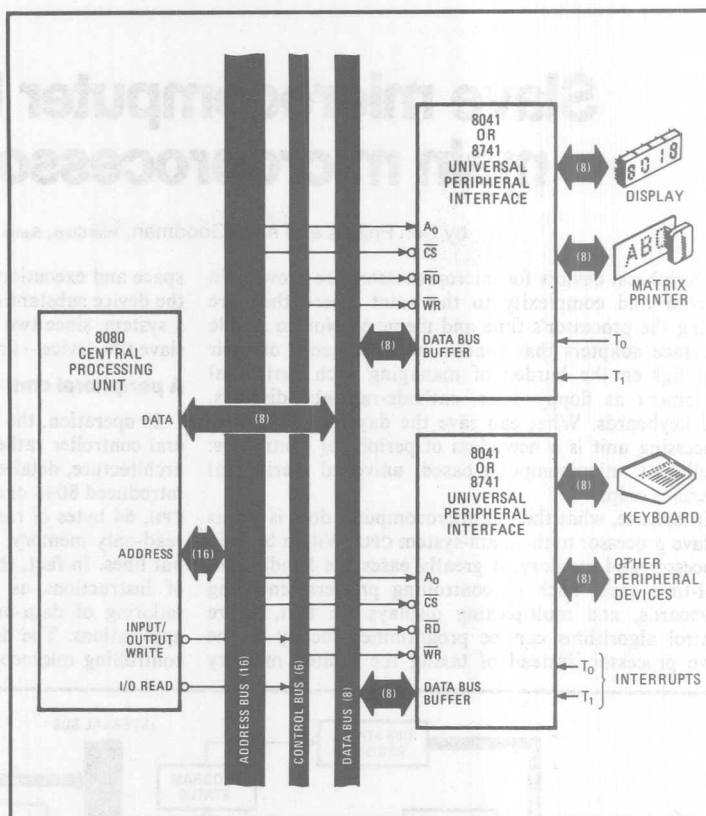


**1. Smart interface.** With an 8-bit CPU, 64 bytes of RAM, and 1,024 words of ROM or erasable PROM, the universal peripheral interface chip is an intelligent peripheral controller rather than a simple interface adapter. The architecture of the chip is similar to that of the 8048 microcomputer. It uses nearly the same instruction set, save for slight variations that improve data-bus operations.

Electronics/July 7, 1977

Reprinted from Electronics Magazine, July 7, 1977. Copyright 1977 McGraw Hill, Inc. All rights reserved.

**2. Slaves.** The microcomputer-based universal peripheral interface chips—the 8741 with erasable PROM and the 8041 with mask-programmed ROM—are connected as slave processors to a main processor (here an 8080 CPU) to take over its I/O chores.



slave processor to the main-system processor.

The chip is the first microcomputer made specifically for a multiprocessor environment in which a master processor sends information to one or more slave processors that in turn control peripheral devices. To accommodate a variety of master processor types, including the 8080, the enhanced 8085, and other 8-bit processors, the chip has bus interface registers that work directly with the central processor's data bus.

Two peripheral controllers are available: the 8741 and the 8041, identical except in one respect. The 8741 has an ultraviolet-erasable, electrically programmable ROM plus the special capability of running through a program a single step at a time. It is designed for low-volume applications requiring program development, as in prototype testing and custom interfacing. The 8041 has a conventional mask-programmable ROM and features a low-power standby mode. It is intended to replace the 8741 once a system design has been set. The 8741/8041 connections for a master-slave arrangement are shown in the block diagram of Fig. 2.

The master processor and the peripheral controller communicate through an asynchronous data-bus buffer register on the UPI. Data and commands are received from the master processor through the DBB, and status and data information are returned through it to the master. The controller sends status information to the

main processor from a 4-bit status register that uses four of the buffer register's eight lines.

The configuration of the DBB and status registers is shown in Fig. 3. The master processor controls data transfer to the UPI by four input lines: the address-input signal specifies whether a command or a data word is being sent; the chip-select line is an enable input that permits communication with the interface, and the read and write lines are used to stroke output and input data, respectively. The master processor uses these signals to direct the exchange of information through the DBB register, which serves as temporary storage for commands and data flowing between master and slave processors.

### The four flags

The status register comprises four flags that direct the handshaking between the master and slave processors. The first is a general-purpose flag, which is set by programming in the 8041/8741 and used to prevent contention over the DBB register between master and slave processors. Another is the command/data flag that, when set, indicates that command information is being transferred. The input-buffer-full flag is set whenever the DBB register is loaded with a word from the main processor, and the output-buffer-full flag is set when the UPI loads its DBB register.

Protocol for the interface begins with the master processor writing an 8-bit character into the buffer register. This sets the IBF flag, signaling the peripheral controller with an internal interrupt. The UPI can then transfer the 8-bit data byte to its accumulator at any time under software control, which clears the IBF flag.

In transferring data in the other direction—from slave to master—the peripheral chip loads the DBB register while automatically setting the OBF flag. The master processor can then read the status register to determine that the OBF flag is set and can proceed to take in data from the buffer register, at the same time clearing the flag in preparation for the arrival of more data.

Transfer of data within the peripheral controller is asynchronous to external processor timing. The chip can thus effectively control peripheral devices while data transfers go on unhindered. Moreover, the DBB register isolates peripheral control tasks from the main processor. Task isolation is desirable in that it eases software development and debugging within a given system (by modularizing functions). In addition, it is certain to enhance data throughput, since two microprocessors are running concurrently.

### Optimized for control

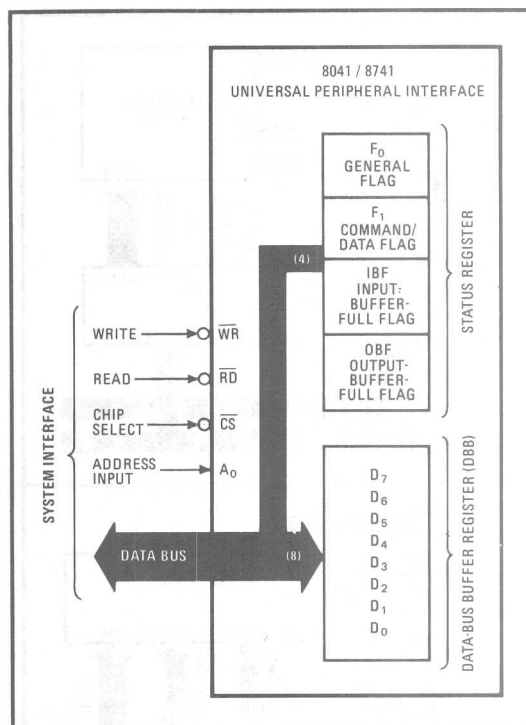
The CPU and instruction set of the 8041/8741 are designed to efficiently handle the single-bit operations required in most control applications, including I/O operations and data-bit manipulation. Two 8-bit-wide ports, compatible with transistor-transistor logic, are provided on the chip. (Sixteen additional lines may be had with the addition of an 8243 I/O expander chip, which takes up half the lines of I/O port 2.) Two inputs to the peripheral controller are provided that may be tested with conditional branch instructions in UPI software. Any port line can be set or cleared individually under software control, and any line can function as either input or output, irrespective of remaining lines.

The timer/event-counter included on the peripheral controller can be preset, read, started, or stopped under software control. In the timing mode, an internal oscillator can be set by a crystal or an LC network. In the event-counter mode, the  $T_1$  input may be used to count switch closures or tachometer pulses, directing program flow accordingly. If the counter has been preset, a flag is available that indicates overflow, and it can signal the master processor.

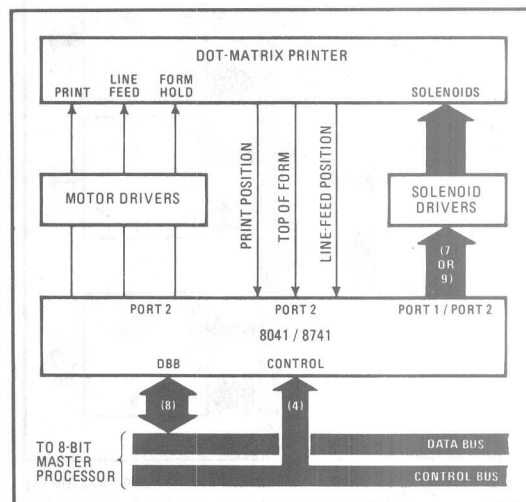
The 1,024 bytes of on-chip ROM are sufficient for most dedicated programming applications. Typically, keyboard encoding or printer control requires 500 to 700 8-bit bytes, and therefore ample program space is available for additional functions.

Of the 64 locations in the on-chip RAM, there are two 8-byte register banks, an eight-level program-counter stack, and 32 bytes of user RAM. The dual 8-byte register banks allow fast response to interrupts such as the IBF flag or time overflow. The stack also provides convenient handling of subroutine cells and storage of other data.

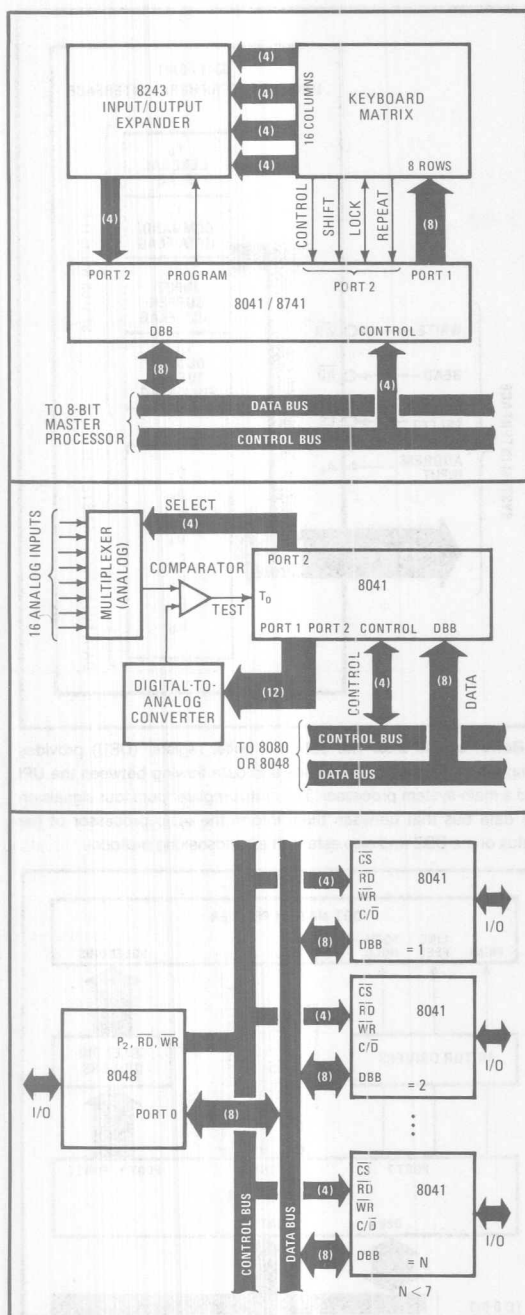
The thrust of the peripheral-controller chip is in its isolation of peripheral tasks from the main processor. Since its job is specifically for control, the main



**3. Buffer to the bus.** The data-bus buffer register (DBB) provides temporary storage for commands and data flowing between the UPI and a main-system processor. The status register puts four signals on the data bus that between them inform the main processor of the status of the DBB and also establish a handshaking protocol.



**4. Printer control.** Memory in the 8741/8041 allows the device to buffer as many as 40 characters to be printed. The main-system processor can transfer a block of data at this speed and then continue with other tasks while the UPI's bidirectional I/O ports monitor and control sequential character printing.



**5. Using the UPI.** Typical applications of the 8741/8041 include (a) a keyboard scanner in which an 8243 input/output expander is added to permit the encoding of as many as 128 keys, (b) a process-control subsystem implemented with an analog multiplexer and a digital-to-analog converter, and (c) a generalized distributed processing system that employs up to seven of the devices as slave processors, connecting them to a single 8048 microcomputer.

processor can therefore be left to down-load commands and transfer data, while the UPI works in real time.

One application might be the controlling of a printer peripheral to an 8080 system, as shown in Fig. 4. The entire real-time control portion of the task can be handled by the peripheral controller. With its built-in timer, it easily handles timing functions like character spacing, print position, and line feed. The UPI has ample I/O ports for a 40-column dot-matrix printer.

In this printer application, the DBB register allows for standardization of data transfer to and from the 8080-based main processing system. To do this, one typical format might be for the main processor to send a start command followed by a full line of 40 ASCII characters. The peripheral controller would then store the characters under program control in a portion of the RAM and begin execution of the printing as soon as the print head and line feed were in the proper position. In the meantime, the main processor returns to other tasks. The ROM in the 8041/8741 can be used to convert the ASCII code to dot-matrix or other formats.

In printer applications, standardization is the key feature offered by the slave peripheral controller. Without any changes in the 8080-based main processing system, the UPI can be programmed to handle any printer mechanisms—dot matrix, drum, spherical head, and so on. In this way, a designer can easily upgrade the peripherals in his system with a minimum of change in the master-processor software.

#### A keyboard application

Figure 5a illustrates an application in which the new chip plus an 8243 I/O expander provide a compact system for scanning and encoding as many as 128 keys from a terminal keyboard. N-key rollover and debounce are implemented by using the on-chip RAM to keep a copy of the key status after each scan. When a key closure is detected, the 8041 uses a ROM look-up table to generate the appropriate ASCII code for transfer to the master processor. As many as 16 characters can be stored and transferred in a block to the master processor.

The analog process-control subsystem illustrated in Fig. 5b can be implemented using an analog multiplexer and digital-to-analog converter along with the 8041. In this configuration, the peripheral controller can monitor and digitize eight analog inputs, perform linearization (using equations or ROM look-up tables), check for limits and zero offsets, and receive control information that could determine new limits.

Figure 5c illustrates a generalized distributed-processing system using as many as seven 8041s as slave processors connected to a single 8048 master processor. Port 2 of the 8048 provides seven chip-select lines to the peripheral controllers plus the command/data control function. This low-cost, low-speed multiprocessor configuration has many advantages over a single high-speed processor. The peripheral controllers are designed especially for control or interface applications, and each can be programmed to handle a single isolated task. This modular approach allows easy development and debugging of the system. □

# MICROCOMPUTER INTERFACING: CHARACTERISTICS OF THE 8253 PROGRAMMABLE INTERVAL TIMER

**Marvin L. DeJong**  
School of the Ozarks

**Jonathan A. Titus and Christopher Titus**  
Tychon, Inc

**Peter R. Rony and David G. Larsen**  
Virginia Polytechnic Institute and State University

As a preliminary discussion, some characteristics of the Intel 8253 programmable interval timer are presented. This extremely versatile input/output chip has various potential uses such as a real-time clock, event counter, and period counter, in addition to replacing software-implemented timing loops. For example, interval timers have been used in a digital cardiometer, a data-logging timer that employed several phototransistors to measure velocities and accelerations, and a program to sample nonperiodic waveforms for subsequent display on an oscilloscope.\*

The 8253 is a 24-pin integrated circuit that requires a single 5-V supply and contains three independent 16-bit interval timers, each of which can be operated in six different modes. An interval timer is a device for measuring the time interval between two actions, or a timer that switches electrical circuits on or off for the duration

\*Dr DeJong of the Dept of Mathematics/Physics at the School of the Ozarks, Point Lookout, Mo has implemented the timers in these simple, but diverse, applications.

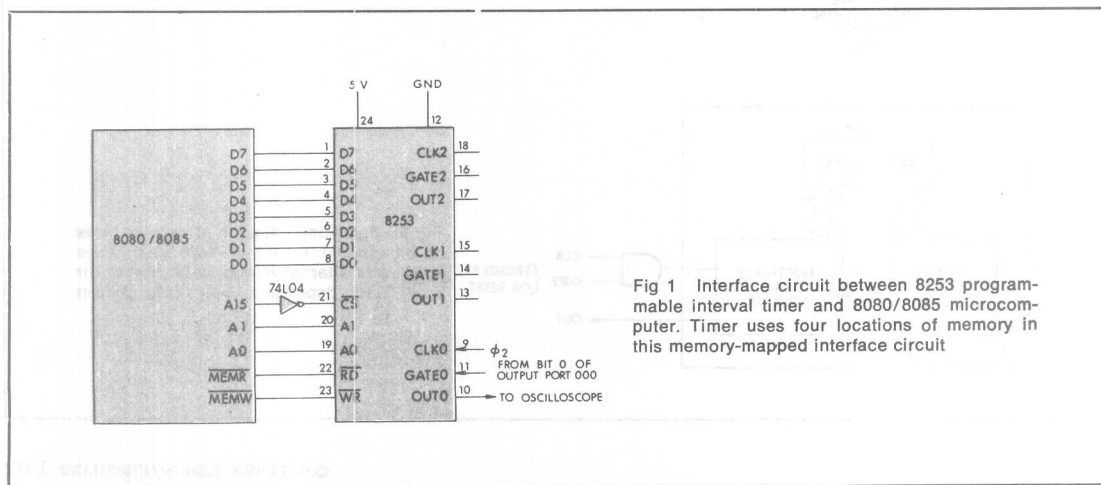


Fig 1 Interface circuit between 8253 programmable interval timer and 8080/8085 microcomputer. Timer uses four locations of memory in this memory-mapped interface circuit

TABLE 1

## Addressing the 8253 Programmable Interval Timer

Control Inputs					Memory Address in Demonstration Program and Interface Circuit
CS	RD	WR	A1	A0	
0	1	0	0	0	Load counter #0 200 000
0	1	0	0	1	Load counter #1 200 001
0	1	0	1	0	Load counter #2 200 002
0	1	0	1	1	Load control register 200 003
0	0	1	0	0	Read counter #0 200 000
0	0	1	0	1	Read counter #1 200 001
0	0	1	1	0	Read counter #2 200 002
0	0	1	1	1	No operation (3-state) —
1	X	X	X	X	Disable chip (3-state) —
0	1	1	X	X	No operation (3-state) —

Note: X = don't care (logic 0 or logic 1)

of the preset time interval.<sup>1</sup> Fig 1 serves the dual purpose of giving the pin diagram of the 8253 chip, while showing how the chip can be interfaced with an 8080A/8085 based microcomputer system using memory-mapped input/output (I/O).<sup>2</sup>

Four internal registers—three interval timers and a control register—that are decoded as memory locations 200 000 through 200 003 with the aid of the address bus signals A0, A1, and A15 (see Fig 1 and Table 1) are contained on the 8253 chip. In Table 1, the RD and WR control inputs determine whether a specific register is being loaded

or read. It is not possible to read the contents of the control register.

Table 2 summarizes the coding for the 8-bit control register within the chip. Bits D7 and D6 determine the selection of the interval timer; bits D5 and D4 determine the nature of the read/write operation associated with the chosen timer; bits D3, D2, and D1, the mode of operation of the timer; and bit D0, whether the timer counts down in binary or binary-coded decimal (BCD).

Fig 2 provides a block diagram for a typical counter in the chip. The microcomputer loads the 16-bit down

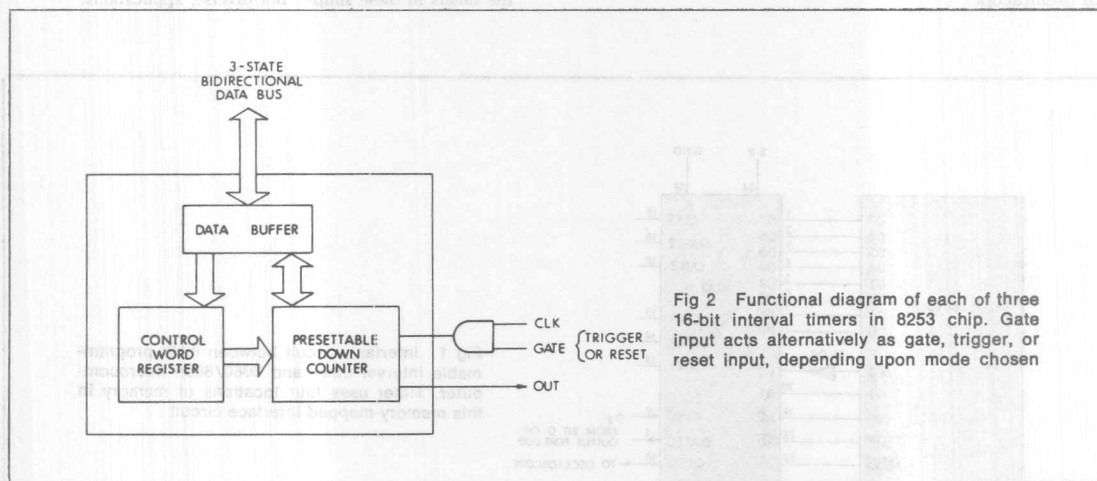


Fig 2 Functional diagram of each of three 16-bit interval timers in 8253 chip. Gate input acts alternatively as gate, trigger, or reset input, depending upon mode chosen

counter as two successive bytes, a HI and LO byte, via the bidirectional data bus, D0 through D7. If the gate line, GATE, is active, negative edge transitions at the CLK input decrement the counter. When the counter reaches zero, OUT becomes active, its actual behavior depending upon the mode programmed into the control register for the counter (see Table 2). The three 16-bit counters on the chip can each be programmed independently in any one of the six modes of operation. Counter inputs and outputs—CLK, GATE, and OUT—for the chosen counter are independent of the CLK, GATE, and OUT i/o of the remaining two counters on the chip.

In addition to the address, data, and control bus connections shown in Fig 1, the CLK0 and GATE0 inputs to counter 0 are respectively connected to the  $\phi 2$  [transistor-transistor logic (TTL)] microcomputer clock output (typically 2 MHz) and to bit 0 of accumulator output port 000. Any TTL level clock with a frequency of less than 2 MHz can be used as input to CLK0, and any suitably debounced switch or source of strobe pulses can be used to control the timer at GATE0. The output of the counter, OUT0, can be connected to an oscilloscope to permit observation of each of the six timer modes of operation.

Next month's discussion will focus on the behavior of a demonstration program for the 8253 programmable peripheral interface chips, which are further described in Refs 3 and 4. This program will illustrate the loading, latching, and reading of counter 0 as well as the various output modes.

**TABLE 2**  
**Coding for 8-Bit Control Register in 8253 Chip**

Bits	Control Function
D7 D6	
0 0	Control word is for counter #0
0 1	Control word is for counter #1
1 0	Control word is for counter #2
1 1	_____

D5 D4	
0 0	Latch both bytes of chosen counter for read operation
0 1	Load or read only most significant byte (MSB) of chosen counter
1 0	Load or read only least significant byte (LSB) of chosen counter
1 1	Load or read LSB first, then MSB of chosen counter
D3 D2 D1	
0 0 0	Mode 0: Output = 1 on zero counter
0 0 1	Mode 1: Retriggerable variable-width one-shot
X 1 0	Mode 2: Programmable rate generator
X 1 1	Mode 3: Programmable square wave generator
1 0 0	Mode 4: Delayed strobe (software triggered strobe)
1 0 1	Mode 5: Triggered strobe (hardware triggered strobe)
D0	
0	Count down in binary
1	Count down in BCD

Note: X = don't care (logic 0 or logic 1)

## References

1. R. F. Graf, *Modern Dictionary of Electronics*, Howard W. Sams & Co, Indianapolis, Ind, 1972, p 298
2. D. G. Larsen, P. R. Rony, and J. A. Titus, *The Bugbook<sup>®</sup> VI. 8080A Microcomputer Programming and Interfacing*, E & L Instruments, Inc, Derby, Conn, 1977, p 21-1
3. *Intel Data Catalog 1977*, Intel Corp, 3065 Bowers Ave, Santa Clara, CA 95051, pp 10-159 (Price, \$2.50)
4. A. Osborne, *An Introduction to Microcomputers, Vol II. Some Real Products*, Osborne and Associates, Berkeley, Calif, 1976, pp 4-106

This article is based, with permission, on a column appearing in *American Laboratory* magazine.

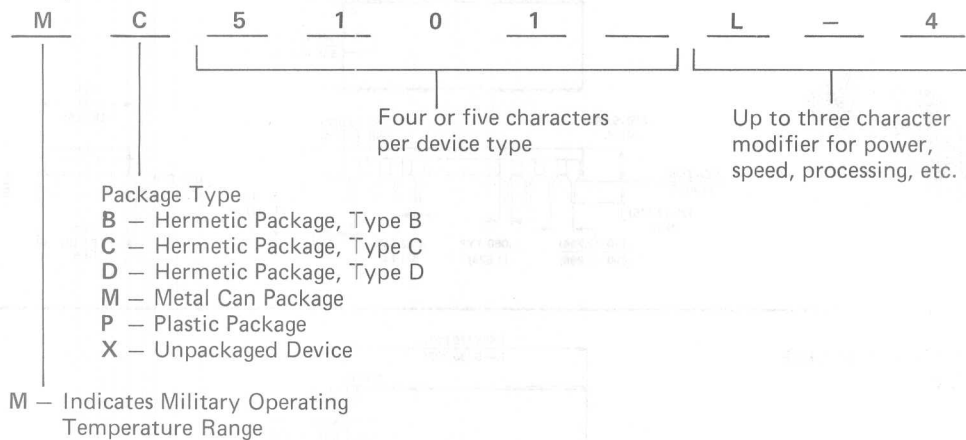




## ORDERING INFORMATION

Semiconductor components are identified as follows:

Example:



Examples:

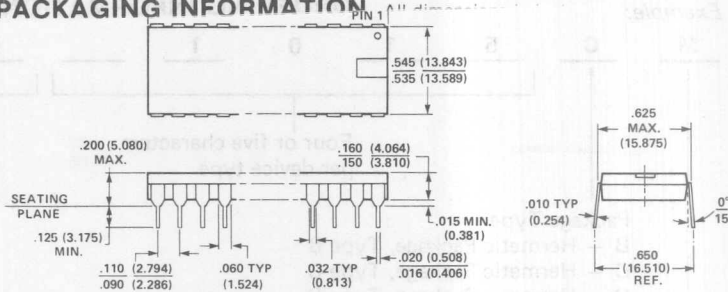
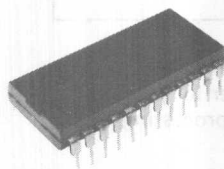
- P5101L** CMOS 256 X 4 RAM, low power selection, plastic package, commercial temperature range.
- C8080A2** 8080A Microprocessor with 1.5  $\mu$ s cycle time, hermetic package Type C, commercial temperature range.
- MD3604/C** 512 X 8 PROM, hermetic package Type D, military temperature range, MIL-STD-883 Level C processing.\*
- MC8080A/B** 8080A Microprocessor, hermetic package Type C, military temperature range, MIL-STD-883 Level B processing.\*

Kits, boards and systems may be ordered using the part number designations in this catalog.

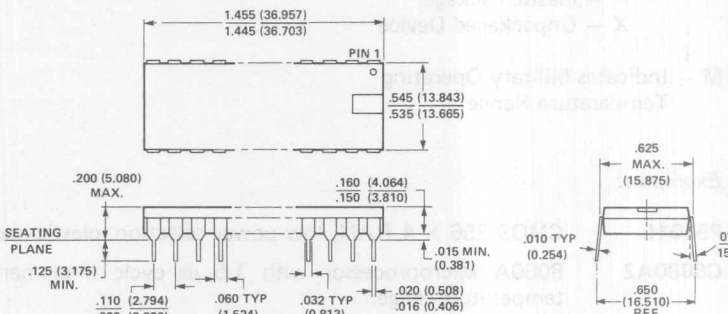
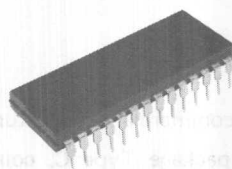
The latest Intel OEM price book should be consulted for availability of various options. These may be obtained from your local Intel representative or by writing directly to Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051.

*\*On military temperature devices, B suffix indicates MIL-STD-883 Level B processing. Suffix C indicates MIL-STD-883 Level C processing. "S" number suffixes must be specified when entering any order for military temperature devices. All orders requesting source inspection will be rejected by Intel.*

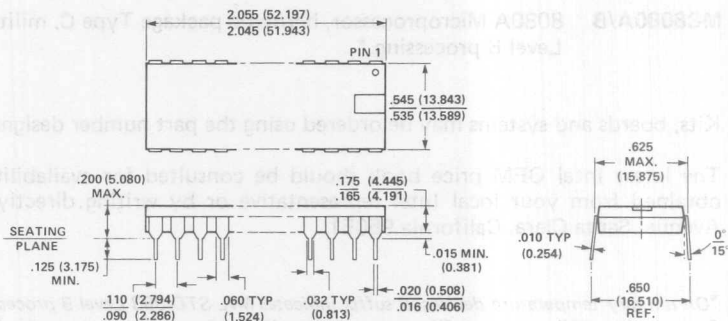
# PACKAGING INFORMATION



28-LEAD PLASTIC DUAL IN-LINE  
PACKAGE TYPE P



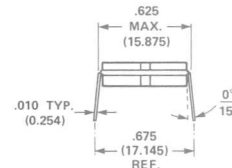
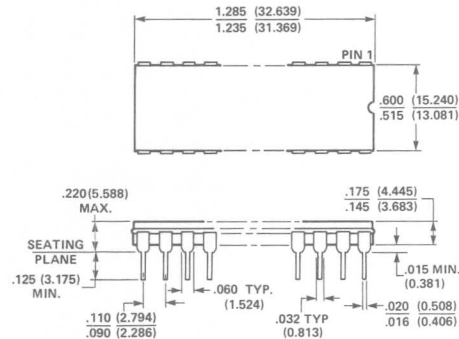
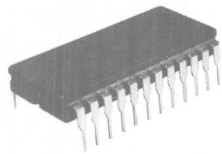
40-LEAD PLASTIC DUAL IN-LINE  
PACKAGE TYPE P



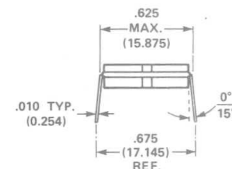
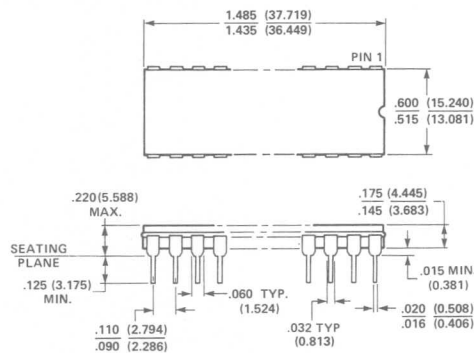
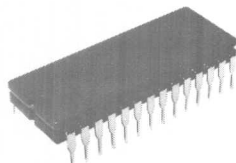
# PACKAGING INFORMATION All dimensions in inches and (millimeters)

## CERAMIC DUAL IN-LINE PACKAGE TYPE D

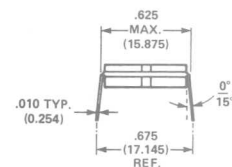
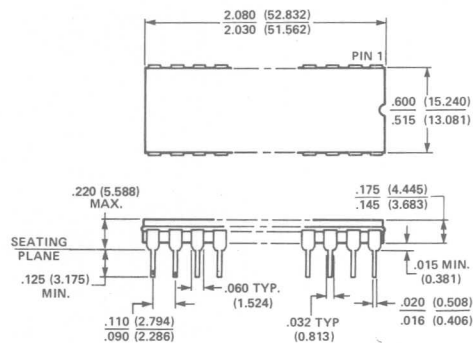
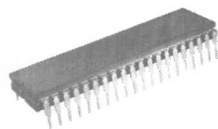
### 24-LEAD HERMETIC DUAL IN-LINE PACKAGE TYPE D



### 28-LEAD HERMETIC DUAL IN-LINE PACKAGE TYPE D



### 40-LEAD HERMETIC DUAL IN-LINE PACKAGE TYPE D



### 40-LEAD HERMETIC DUAL IN-LINE PACKAGE TYPE B

